THESIS

PERFORMANCE ANALYSIS FOR A VISION-BASED TARGET TRACKING SYSTEM OF A SMALL UNMANNED AERIAL VEHICLE

by

Todd Michael Trago

September 2005

Thesis Advisor:	Isaac Kaminer
Second Reader:	Vladimir Dobrokhodov

REPORT DOCUMENTATION PAGE	*Form Approved OMB No. 0704-0188*

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.

1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** September 2005	**3. REPORT TYPE AND DATES COVERED** Master's Thesis

4. TITLE AND SUBTITLE: Performance Analysis for a Vision-Based Target Tracking System of a Small Unmanned Aerial Vehicle	**5. FUNDING NUMBERS**
6. AUTHOR(S) Todd Michael Trago	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000	**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A	**10. SPONSORING/MONITORING AGENCY REPORT NUMBER**

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION / AVAILABILITY STATEMENT **Approved for public release; distribution is unlimited**	**12b. DISTRIBUTION CODE**

13. ABSTRACT (maximum 200 words)

This thesis analyzes performance of the vision-based target-tracking system developed at the Naval Postgraduate School using the Monte Carlo method. Specifically, sensitivities of the target position estimation algorithm to various sensor errors are computed and analyzed. Furthermore, dependence of this algorithm on the performance of the target-tracking control system is established.

14. SUBJECT TERMS Small Unmanned Aerial Vehicle, Vision-Based Target-Tracking, Monte Carlo Method, Range Estimation, Piccolo, Graceful Degradation of Performance	**15. NUMBER OF PAGES** 87
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT** UL

NSN 7540-01-280-5500 Standard Form 298 (Rev. 2-89) Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

PERFORMANCE ANALYSIS FOR A VISION-BASED TARGET TRACKING SYSTEM OF A SMALL UNMANNED AERIAL VEHICLE

Todd M. Trago
Ensign, United States Navy
B.S., The Citadel, 2004

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN ENGINEERING SCIENCE (MECHANICAL ENGINEERING)

from the

NAVAL POSTGRADUATE SCHOOL
September 2005

Author:	Todd Michael Trago
Approved by:	Isaac Kaminer Thesis Advisor
	Vladimir Dobrokhodov Second Reader
	Anthony J. Healey Chairman, Department of Mechanical and Astronautical Engineering

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

This thesis analyzes performance of the vision-based target-tracking system developed at the Naval Postgraduate School using the Monte Carlo method. Specifically, sensitivities of the target position estimation algorithm to various sensor errors are computed and analyzed. Furthermore, dependence of this algorithm on the performance of the target-tracking control system is established.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

First and foremost, I would like to thank Dr. Isaac Kaminer for giving me the chance to work with him on this project. It has proven to be the most fulfilling academic experience in my life thus far. Next, proper thanks must be given to Dr. Vladimir Dobrokhodov. His innumerable hours of help and explanation of the various aspects of this project have been invaluable to me in many ways. I would also like to thank Dr. Kevin Jones for his constant work on the physical systems of the SUAV both in the lab and at McMillan airfield.

On a more personal and intimate level, everything that I have accomplished here at NPS is because of my fiancé Allison. If it weren’t for her continuous love and support, none of this would be possible. I love you Allie.

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF ACRONYMS

NPS	Naval Postgraduate School
SUAV	Small Unmanned Aerial Vehicle
GPS	Global Positioning System
IMU	Inertial Measurement Unit
VBTT	Vision-Based Target Tracking
AMT	Automatic Target Tracking
RMS	Root-Mean Square
HITL	Hardware-in-the-Loop
CRC	Cycle Redundancy Check
GNC	Guidance Navigation and Control
LOS	Line of Sight

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

A. BACKGROUND

The mission of the Small Unmanned Aerial Vehicle (SUAV) developed at the Naval Postgraduate School (NPS) is to support the Tactical Network Topology (TNT) project. The TNT experiments are being performed by the NPS in conjunction with USSOCOM. The overall mission of the TNT project is to explore and develop warfare capabilities for network-based entities.

The function of the image-based target tracking operator of the modern UAV complex is to acquire a target, continuously track this target, determine its position based on real time video and available telemetry, and finally to relay this information across the network. From the real time video information taken on the target, important tactical decisions can be made.

B. PROBLEM STATEMENT

The purpose of this thesis is to perform numerical sensitivity analysis of the target position estimation system of the NPS SUAV. To this end simulation experiments were created and ran by using the basic principles of the Monte Carlo method. The experiments were conducted to acquire information on errors in the NPS SUAV system, then analyze the contribution of the multiple sources of the errors. The first step taken in order to successfully run these simulations was to study the NPS SUAV project and become familiar with its hardware and software. The second step was to set up and run the Monte Carlo simulations. Finally, the third step was to acquire the proper information from the simulations and analyze the results.

THIS PAGE INTENTIONALLY LEFT BLANK

II. NPS UAV PROJECT OVERVIEW

A. SYSTEM DESCRIPTION

The following information on the different components of the NPS SUAV was taken from personal experience, the other members of the SUAV project, and [Ref. 1].

1. NPS SUAV

The NPS SUAV is a Senior Telemaster RC aircraft (shown in Figure 1. below), modified to suit the projects needs. It houses the gimbaled camera, wireless video and serial links as well as Piccolo autopilot with its dedicated control link. The SUAV has a 2.5m wingspan, and 8kg weight is heavily altered to suit our needs. These alterations include a reinforced two-piece bolt-on wing with barn-door ailerons, removable tail surfaces, and a much larger fuselage to house payload components. The SUAV is powered by a 23cm^3 two-stroke gasoline engine with a 1500cm^3 gas tank, giving it an endurance of about 3 hours in the air.

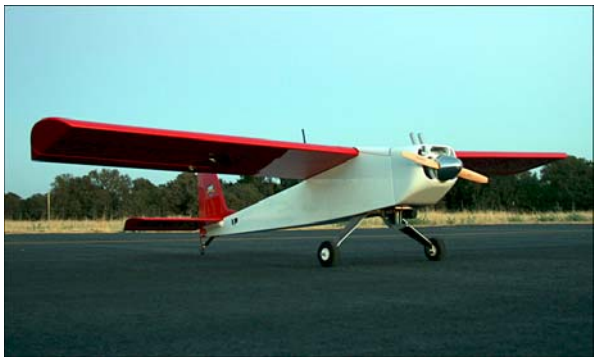

Figure 1. The NPS SUAV

The SUAVs main payload is a gimbaled camera, mounted just beneath the nose. The electronics bay, which holds the majority of the SUAVs electronic equipment, is found on the aft fuselage. The bay is made up of lightweight aircraft-plywood trays which slide into EPT foam blocks, allowing for vibration isolation and easy access to all of the components. To improve the lifespan of the electronics and the video quality, a Hyde softmount is used to isolate engine vibration from the airplane chassis. The equipment housed in the bay includes slide-in cards with the video transmitter, a serial-to-PWM card to drive the gimbal servos, and a Freewave radio to receive the serial commands for the gimbal and the power conversion for the servos from the ground. Figure 2 below shows the SUAVs electronics bay.

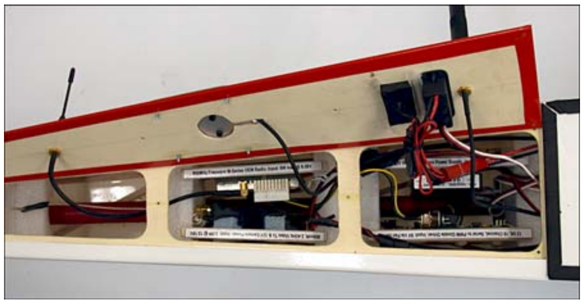

Figure 2. The SUAVs electronics bay

2. Piccolo Autopilot

All of the following information on the Piccolo autopilot and its components are taken from [Ref. 1] and [Ref. 6].

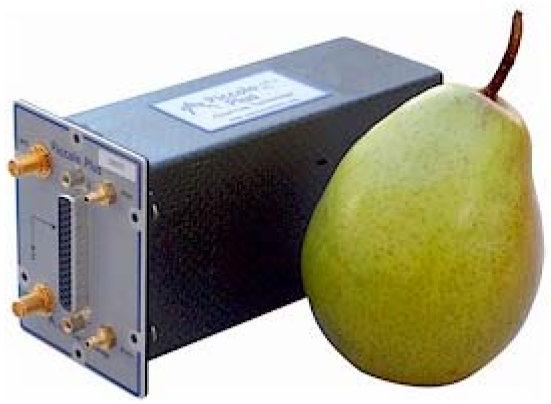

Figure 3. The Piccolo Plus Autopilot (From: [Ref. 6])

a. Control System

The NPS SUAV control system is made up of four separate parts. The first of the four is the Piccolo avionics control system, which is attached onboard the SUAV. This part of the control system will be covered in detail in a following section. The second part of the control system is the Piccolo ground station. This will also be discussed in detail in the ground station section of this thesis. The third part of the control system is the computer used for Operational Interface (OI). A person manning the OI can control all stages of the SUAV life by manipulating various system parameters implemented in the user interface. The final part of the control system is manual control interface. This consists of a pilot with a controller who takes command of the SUAV during take-off, landing, and in times of emergency.

The control system is based on traditional inner and outer loop configuration, and therefore is of two separate loops. The first inner-loop, which is implemented inside the Piccolo autopilot, is fast and controls the SUAVs flight dynamics. The second outer-loop is slower and controls the navigation tasks; it resides in the ground station that communicates with the Piccolo autopilot through a wireless RF link.

b. Avionics Sub-System

The Piccolo avionics sub-system is the only system onboard the SUAV. It is intended to act as the autopilot by acquiring commands from the ground station and provides control commands to the surfaces of the SUAV, which include the throttle, elevators, rudder, and ailerons.

c. Avionics Processor

The processor included in the Piccolo avionics sub-system as well as in the Ground Station is a MPC555 microcontroller capable of 40MHz of PowerPC operation. It connects many interfaces to a powerful RISC engine. This CPU controls everything in the Piccolo avionics sub-system, and generates a moderate amount of data filtering, allowing for more precise results.

d. Accelerometers

The Piccolo autopilot is outfitted with two Motorola ADXL202 accelerometers. These accelerometers are low-power, low-cost and allow for a measurement range of plus or minus 2g sensitivity. The two-axis accelerometers allow for measurement of acceleration in the X, Y, and Z direction. These measurements are set to the microprocessor, and then filtered along with other measurements from the gyros.

e. Gyros

The Piccolo autopilot is outfitted with three Tokin CG16D rate gyros, which can be placed at any attitude due to the CPUs durability.

f. Pressure Sensors

The Piccolo autopilot is outfitted with three different pressure sensors. The first is an MPXV 4kPa dual ported dynamic pressure sensor. The second is an

MPX4115a absolute ported barometric pressure sensor. Finally, the third is a temperature sensor. Through the combined use of these three pressure sensors, the CPU can calculate altitude and true airspeed.

g. GPS

The Piccolo autopilot is outfitted with a Motorola M12 GPS reciever. The Piccolo ground station is also equipped with the same Motorola M12 GPS sensor. This device provides the standard GPS measurements of the SUAV.

h. Ground Station

The Piccolo ground station hardware is a copy of the Piccolo autopilot hardware. The ground station administers the RF communications with the Piccolo autopilot, and gives the operator interface a constant data stream of control data from the autopilot. The pilot, who is used for manual control of the UAV, is brought into the loop here as well. A control switch in the ground station gives command of the SUAVs controls to the pilot.

i. Operator Interface

The Piccolo operator interface consists of a laptop computer that executes the Piccolo software. It is easily manned by one person, and requires a person with knowledge of the software, and the flight plan to be performed. Some of the many features that the interface includes are flight planning in the form of setting up way points to fly, and calibration of flight sensors.

j. Manual Control

Manual control of the SUAV at the airfield is done either by a hired pilot or Dr. Jones. Through utilization of the Piccolo operator interface in the Piccolo ground station, the pilot is given control of the SUAV. This process is easy and instantaneous with the flip of a switch. While in manual control mode, the pilot flies the SUAV with a Futaba controller that talks directly to the Piccolo ground station.

k. Communications Protocol

The communications between the Piccolo autopilot and Piccolo ground station are encoded in a distinctive two-layer protocol. The outer layer consists of information on the specific avionics equipment sending the data, payload size, and additional information from the inner layer of the protocol. The inner layer consists of information on the type and size of the data to be sent. This is then followed by the actual data itself. In order to guarantee that the data matches and no errors in communications occurred, synchronization bytes are used.

l. Piccolo Plus Simulator

A very useful feature of the Piccolo autopilot is that it has a powerful hardware-in-the-loop (HITL) simulator included with it. This simulator allows the user to fully test mission functionality and control laws without any risk to hardware. The use of this simulator is very important to the NPS SUAV project, because it greatly decreases the likelihood of failure at the airfield by finding any deficiencies in the lab. The HITL simulator is based on an external CAN interface. The CAN interface acts as a bus between the Piccolo autopilot and the computer with a CAN interface card that hosts the simulator of an aircrafts flight dynamics. The Piccolo autopilot sends servo control information and attains external sensor information via the CAN bus.

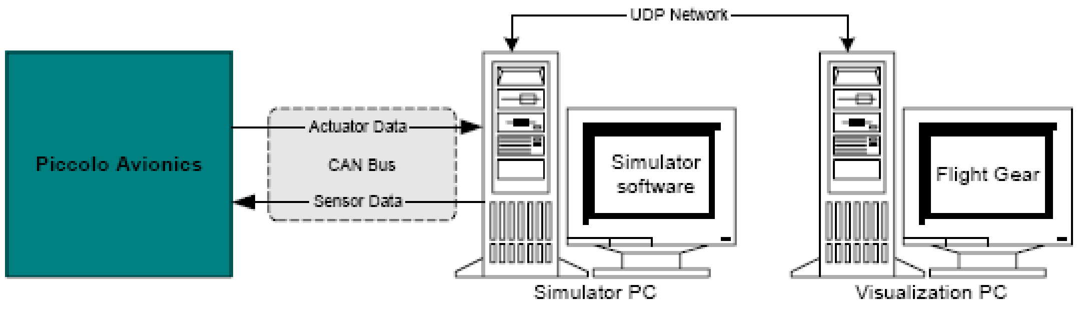

Figure 4. HIL Simulation. (From: [Ref. 6])

The HITL simulator has a flight simulation and visualization program called Flight Gear. This program can be used to see and follow the SUAV as it is flying in the simulation.

FlightGear
File View Location Autopilot Weather Equipment Debug ATC/AI Help
40.0

Figure 5. Flight Gear Flight Simulation

3. Gimbaled Camera

The gimbaled camera consists of a custom pan-tilt unit is driven by high-end COTS hobby servos. It is controlled from the ground with a 900MHz serial modem, providing it with 360º of pan, and 90º of tilt. Video is transmitted to the ground using an 800mW 2.4GHz link from Electra Enterprises. This link is made up of an omni antenna, which gives a range of a few kilometers. The gimbal currently has a high resolution, low-light, black and white camera. However, it is designed to accept two cameras at a time, and eventually either a color camera or an IR camera will be added.

4. NPS Ground Station

The NPS ground station acts as the SUAV projects ground control base. It is the vital link that holds the entire project together, and includes the Piccolo autopilot/ground station/simulator, the PC-104/xPC Target computer, the PerceptiVU host computer, and the NPS ground station control.

a. PC-104/xPC Target

The xPC target and Real Time Workshop software was created by The Mathworks Inc., and permits models made in Simulink to be downloaded and ran on a target computer. The target computer is a stackable PC-104 computer system which runs the models in real-time during flight of the SUAV.

b. PerceptiVU Image-Tracking Software

The PerceptiVU VBTT software was created by PerceptiVU Inc., and takes analog video from the pan-tilt camera unit on board the SUAV. A user, manning this part of the ground control station, manipulates a video game joystick which controls the pan-tilt camera on board the SUAV. The user can acquire a target by clicking the trigger button on the joystick. Once a target is acquired, the camera unit will stay locked on that target until an error occurs, or the user cancels the target to attain another. When tracking a target, the PerceptiVU software will yield the targets off-set position.

The PerceptiVU software is very flexible, allowing the user many different options for making the acquisition of a target fast and easy. Options such as black hot and white hot polarity allow the user to change the sensitivity of the software to either black or white targets. This makes it easier to track black targets in lighter surroundings or white targets in darker surroundings.

B. FLIGHT CONTROL

The NPS SUAVs mission and flight control is done using the Piccolo autopilot system, and all of its components as described in section A above. A basic description of these four parts is also shown above in section A. These four parts combine to provide a very reliable means for flying a SUAV. Figure 6 below shows the basic control system set-up of the NPS SUAV.

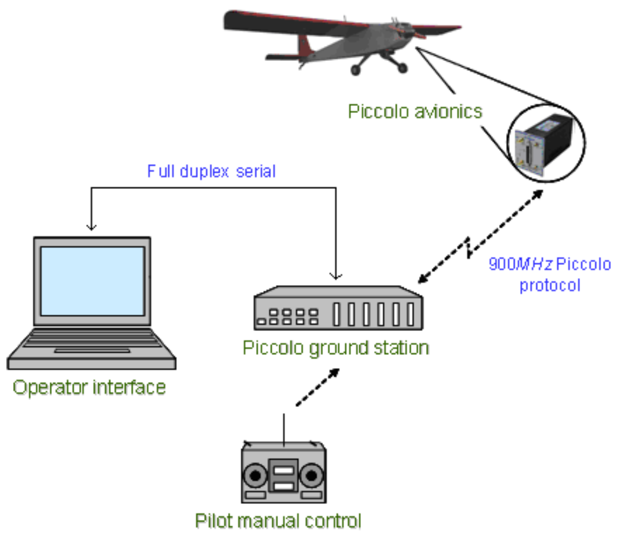

Figure 6. NPS SUAV Control System Set-Up

In Figure 7 shown below, it is seen that the Piccolo control set-up is made up of two control loops. The first is a faster inner loop which is executed by the Piccolo autopilot. This inner loop was developed by NPS and implemented on Piccolo by Cloud Cap, thus providing NPS the ability to test algorithms developed in-house. The second is a slower loop, which is implemented by the NPS ground station.

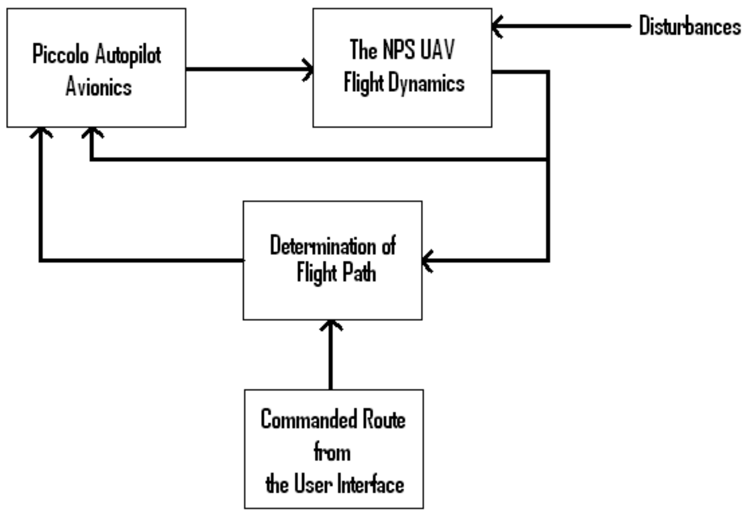
Piccolo Autopilot Avionics
The NPS UAV Flight Dynamics
Disturbances
Determination of Flight Path
Commanded Route from the User Interface

Figure 7. The Piccolo Control System (Closed Loop)

Communication between NPS, the Piccolo ground station, and the Piccolo autopilot consists of an application-specific two-layer protocol, which communicates between the systems components. This two-layer protocol can be seen below in Figure 8. The inner layer of the protocol is made up of two synchronization bytes, the payload itself, and finally a two byte cycle redundancy check (CRC). The first of the two synchronization bytes consist of one byte to denote the type of the incoming message, whether it be waypoint, telemetry, control data, or any of the 24 data types communicated by the control system. The second synchronization byte denotes the size of the coming payload. The CRC code is implemented to spot data transmission errors. The outer layer of the two-layer protocol is made up of two synchronization bytes, two network address bytes, one byte stream identifier, one byte to denote the size of the coming payload, the payload itself, one byte containing an eight bit XOR checksum of header bytes, and a two byte CRC code.

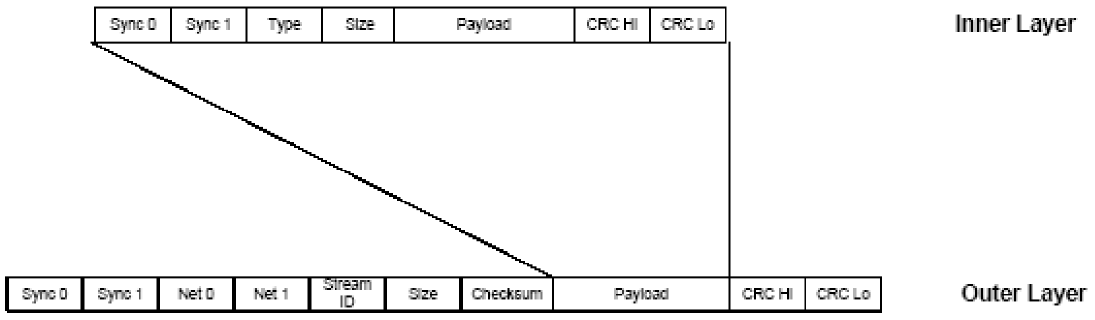

Figure 8. Two-Layer Protocol of Piccolo Communications (From: [Ref. 2])

C. TARGET TRACKING

The NPS UAV project uses a Vision-Based Target Tracking (VBTT) system, shown in Figure 9, which acquires a user designated target, and gets an estimate of this targets GPS position.

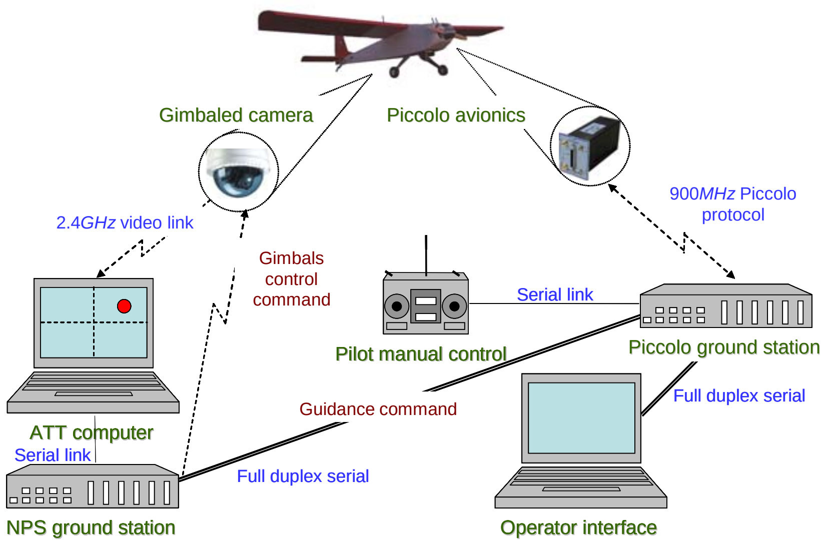

Figure 9. The Vision-Based Target Tracking System

During a flight test at Camp Roberts, the team member positioned at the ATT computer can locate and designate a target. The target will then appear in a rectangular box, which goes from being yellow to being red when designated.

The UAV GNC algorithm, shown in Figure 10 below, provides coordinated control of the SUAV and the gimbaled camera to keep the designated target in the center of the camera frame. It also estimates the targets position.

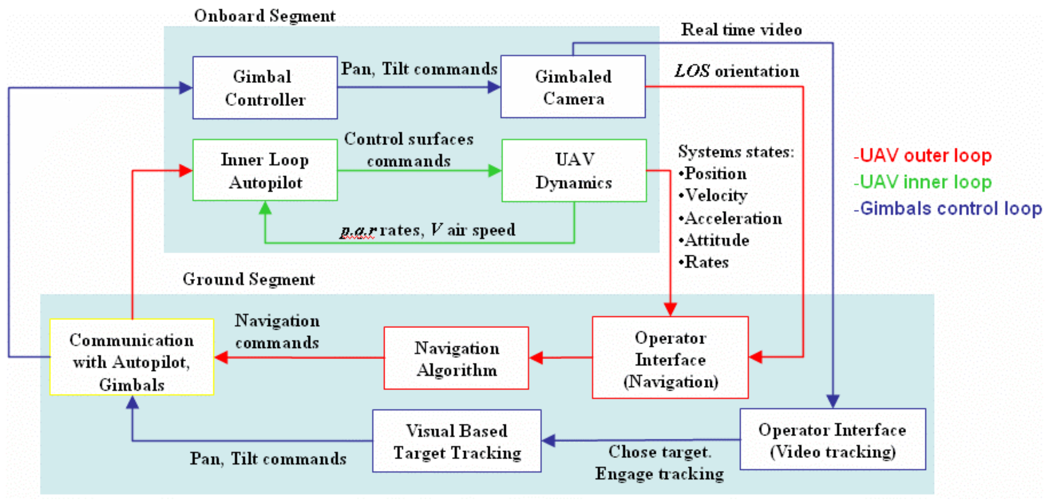

Figure 10. The Guidance Navigation and Control (GNC) Algorithm

The target position is obtained by using two estimation filters. The first filter is a Polar coordinate based estimator, which is a constant gain Kalman filter for the line of sight (LOS) rate and range estimation. The second of the two filters is a Cartesian coordinate based estimator.

D. ERRORS IN TARGET POSITION ESTIMATION

1. Introduction

Many different components of the NPS UAV system can contribute to the overall target position error. These sources of errors and how they can affect the total system error are discussed in the following sections.

2. Gimbaled Camera

a. Positioning Errors

Errors in the measurement of the gimbaled camera angles drastically affect the performance of the estimation filters. The gimbaled camera is manipulated via a gimbal control algorithm. Any noise that could affect this algorithm would in tern affect the performance of the VBTT system as a whole.

3. Video Quality

The resolution of the camera can be a source of error contributing to the total system error. Resolution can be defined as the clarity and sharpness of an image. Many things can affect the cameras resolution, and in turn cause errors. Some of these sources include the actual camera lens, sensor design, interpolation algorithm, focus distance, vibrations, scene contrast, and position in the image field. Two of these sources that have the greatest influence on the camera resolution of this project are definitely vibrations and scene contrast. When these two factors come into play the camera resolution is affected, and PerceptiVU may lose target lock.

When performing flight tests on the NPS SUAV at Camp Roberts, vibrations caused by the engine consistently affect the quality of the video. Contrast also has a drastic affect on the quality of the video. On very bright days in the spring and summer times at Camp Roberts, scene contrast plays a significant role in producing errors for the system. During these days, glare from the sun on the bright colored grass, and the color of the grass itself greatly hamper the camera resolution by causing white out. From the air, the objects are indistinguishable when the white-out takes place. Figures 11 and 12 located below show the affects of grass color and glare on camera resolution. In Figure

11, an ideal camera frame is shown. In this frame, the target is easily distinguishable from its surroundings, making it easy to acquire and maintain. Figure 12 shows a camera frame in which white-out occurred, and almost no distinction can be made between a target and its surroundings. As stated above, when almost no distinction between objects can be made target acquisition can be close to impossible.

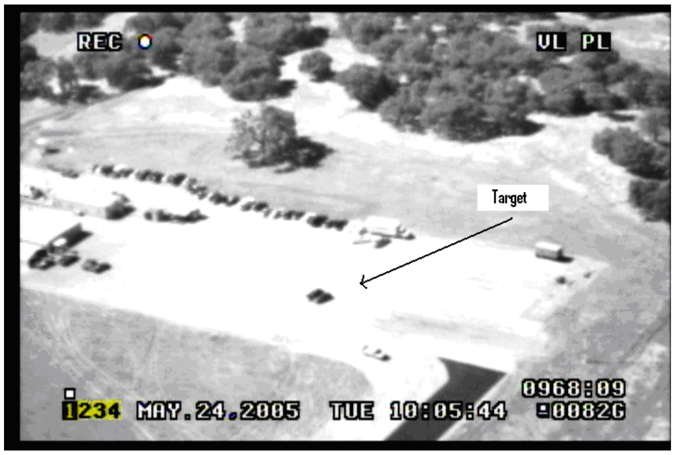

Figure 11. An Ideal Camera Frame

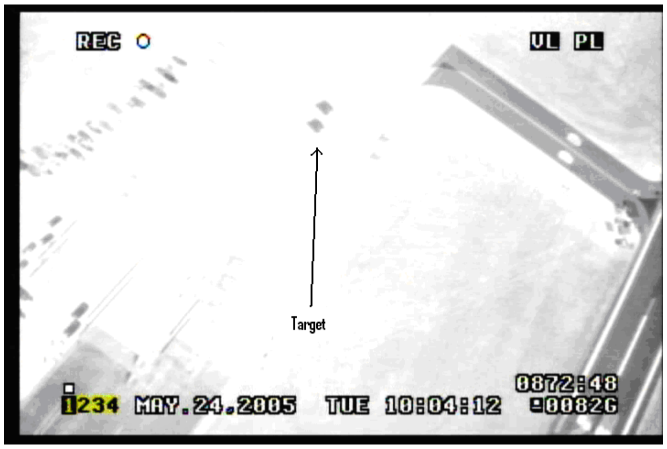

Figure 12. White-out Camera Frame

E. THE MONTE CARLO METHOD

1. Introduction

The Monte Carlo method can be defined as a technique of sampling a particular distribution by use of random numbers. Because of its use of random or pseudorandom numbers, the Monte Carlo method is sometimes referred to as stochastic simulation.

The Monte Carlo method can be easily used in many fields. Because of the Monte Carlo methods ease in helping people solve problems of exponentially increasing complexity and computational effort, its range of application is broadening. One can plainly see why the Monte Carlo method is presently one of the most robust and commonly used practices for solving a complex problem [Ref. 3].

The Monte Carlo methods used in this thesis will greatly benefit the NPS SUAV project. Through use of the Monte Carlo method, the NPS SUAV team will have better insight into how the SUAV acts in response to random noises. The Monte Carlo method will be implemented by introducing random noise into the control system that governs the UAVs flight, VBTT capabilities, and target range estimation filters. Figure 13, located below, shows a straightforward representation of what the Monte Carlo method as implemented into this system will hope to accomplish.

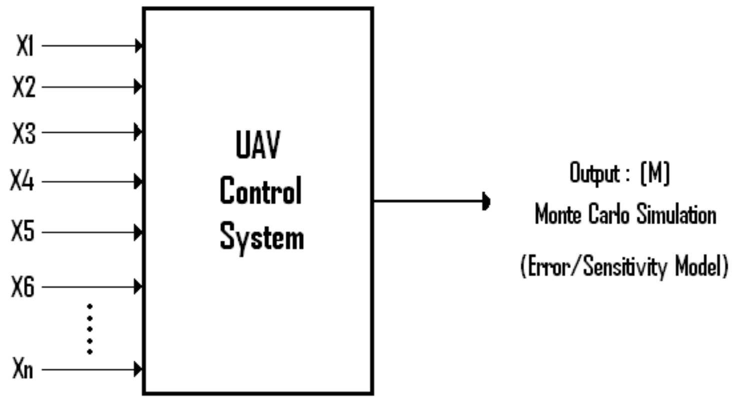

Figure 13. Simple Diagram of the Monte Carlo Simulation

In Figure 13, the X variables on the right of the diagram represent inputs. These inputs represent the sensor measurements with random white noise added to them to simulate random error. This causes the system to deviate from nominal performance. This is the Monte Carlo method of simulation. The output will be comprised of an error and sensitivity matrix M, that shows how many of the SUAVs sensors were analyzed. The output of the simulation is explained through the equations below. In the matrix below, M_K represents different model performances. The model performances that will be analyzed include range estimation, guidance, and target tracking. X_i symbolizes the different error iterations used in the simulation. This includes different mean and variance values given to a white noise signal which is input into the control system.

$$M = \left[\frac{\delta M_K}{\delta X_n}\right] = \begin{bmatrix} \frac{\delta M_1}{\delta X_1} & \frac{\delta M_2}{\delta X_1} & \cdot & \cdot & \cdot & \frac{\delta M_K}{\delta X_1} \\ \frac{\delta M_1}{\delta X_2} & \frac{\delta M_2}{\delta X_2} & \cdot & \cdot & \cdot & \frac{\delta M_K}{\delta X_2} \\ \cdot & \cdot & \cdot & & & \cdot \\ \cdot & \cdot & & \cdot & & \cdot \\ \cdot & \cdot & & & \cdot & \cdot \\ \frac{\delta M_1}{\delta X_n} & \frac{\delta M_2}{\delta X_n} & \cdot & \cdot & \cdot & \frac{\delta M_K}{\delta X_n} \end{bmatrix} \tag{1.1}$$

The Monte Carlo method will be done using Matlab and Simulink. This method of simulation will be discussed in further detail in the next section of this thesis.

2. Statistics Review

The intention of this section of the thesis is to give a brief review of probability, statistics, and the mathematical properties used in simulating the NPS SUAV. For a more in depth look into these topics, see [Ref. 4]-[Ref. 5].

1. Probability and Statistics

The mathematical information provided below was taken from [Ref. 5]. If one is given some random occurrence A, with N random outcomes $A_1, A_2, A_3, ...A_N$, there is a probability P_K between 0 and 1 associated with each outcome A_K.

$$0 \le P_K \le 1 \tag{1.2}$$

A common notation for the probability of random occurrence A_K is

$$P_K = p(A_K) \tag{1.3}$$

For each outcome A_K, there is a numerical value X_K associated with it. The function assignment X_K to A_K is called a random variable. A random process is made up of a set of random variables indexed by time. The expected value $E(x)$ of a random variable X_K is described as

$$E(x) = \sum_i P_i X_i = \mu \tag{1.4}$$

The expected value of a random variable can also be considered the statistical mean of a random variable. The nth central moment of X_K is defined as the expected value of the nth power of X_K

$$E((X-\mu)^n = \sum_i P_i (X_i - E(x))^n \tag{1.5}$$

The variance of X_K is called the second moment of X_K, and is given by

$$E((X-\mu)^2) = E((X-E(x))^2) \tag{1.6}$$

One can obtain the equation for the standard deviation σ of X_K from (1.6) as well. The standard deviation of X_K is a measure of the distribution of the random variable from the expected value. The equation for the standard deviation is,

$$\sigma = \sqrt{E((X-E(x))^2)} = \sqrt{E(x^2)-(E(x))^2} \tag{1.7}$$

The root-mean squared (RMS) is an arithmetical measure of the scale of a changing quantity. RMS values can be calculated for either a continuously varying function or a series of discrete values. The equation for the root-mean square is,

$$X_{RMS} = \sqrt{\frac{1}{N}\sum_i^N ({X_i}^2)} = \sqrt{\frac{{X_1}^2 + {X_2}^2 + \ldots + {X_N}^2}{N}} \tag{1.8}$$

A continuous random variable X_K that is normally distributed can be graphically characterized by a bell curve. This bell curve has the following probability density function (PDF),

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e \left| \frac{(X-\mu)^2}{2\sigma^2} \right| \tag{1.9}$$

The random noise propagates through the VBTT system does so in a nonlinear way. Therefore, only standard deviation and RMS values can be computed and used to describe the errors in the NPS SUAV.

III. SIMULATION IMPLEMENTATION

A. SIMULINK IMPLEMENTATION

1. Introduction

The Monte Carlo simulation was implemented via using Simulink. This was done introducing white noise to certain sensors of the model. Figure 14 below shows the Simulink model used for the implementation of the Monte Carlo method.

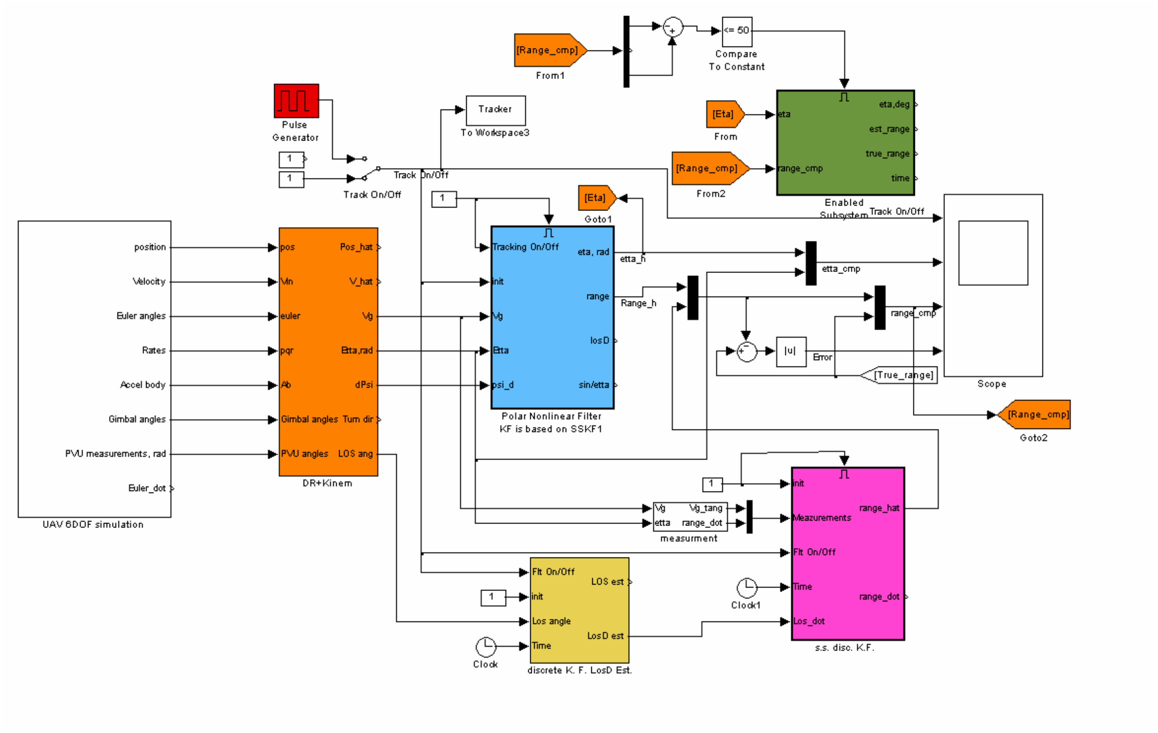

Figure 14. Developed Simulation Model, Sim3

To successfully run the Monte Carlo method on this model, sensors had to be chosen from the model to introduce noise to. These sensors are important to the performance of the SUAVs VBTT system, including its position estimation capabilities. Nine total sensors were chosen for the Monte Carlo implementation. The following sensors were chosen to be manipulated in order to find the sensitivity of the range estimation capability of the VBTT system. The first eight parameters include: pan and tilt of the gimbaled camera; Euler angles and rates. The last of the nine parameters is the Boolean signal of variable duration that models the target loss event (see the red block in Figure 14); this signal is similar to the standard pulse width modulation one. The manipulation of this parameter will also be done for sensitivity calculations of the target-tracking and range estimation models. After the white noise is introduced to the system, analysis will be done, and a sensitivity matrix for the system will be obtained. Included in this sensitivity matrix will be how the sensors performed with the white noise input into the system.

Before all the noise can be implemented into the system however, a few other simulations must be run. First, an ideal simulation will be executed. In this scenario, no errors will be introduced into the system, so that one can see how the NPS SUAV will perform under nominal conditions. This is also done so that once noise is introduced into the system there will be a nominal model to compare the sensitivity models to. After the ideal simulation has been run, sensitivity analysis will be performed for all of the chosen sensors individually. The purpose of doing this is to see how each individual sensor affects the position estimation performance of the VBTT system. More importantly, it will show which sensors affect the position estimation error the most so that the SUAV team will know what specific areas of the SUAV must be improved upon. This entails enabling one sensor at a time and adding the white noise to it. After all nine sensors are simulated they will be statistically analyzed and compared to each other with respect to the position estimation capabilities of the VBTT system. This comparison will be the sensitivity model of the NPS SUAV. Data will also be taken from the simulation in the form of the result matrix shown above.

Another small but important simulation will be run before every noise parameter is added. This simulation will test how the target-tracking ability of the SUAV affects its position estimation capabilities. For this simulation, a pulse width generator discussed above will be manipulated. More on this simulation will be discussed in the following sections. Once these three simulations have been run, all noise channels will be turned on and simulated to the model. The same scenario will be run many times with different amounts of errors introduced into the system. Once this is done, a sensitivity model for the position estimation performance will be obtained. The main focus of this sensitivity model will be the guidance angle eta η- the angle between the SUAVs ground velocity vector and the perpendicular of the LOS vector. The figure displayed below shows how η is related to the basic kinematics model of the circular guidance algorithm.

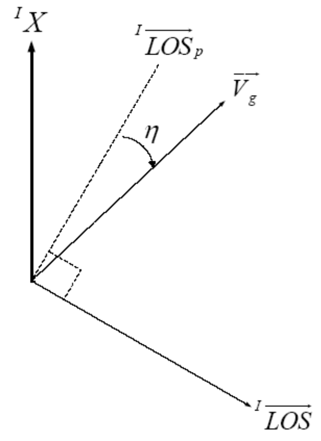

Figure 15. Angle Relationships for a 2-D Kinematics Model (After: [Ref. 1])

The equation below shows the mathematical representation of η.

$$\eta = \sin^{-1}\frac{\left\| {}^{I}\overrightarrow{LOS_P} \times \overrightarrow{V_g} \right\|}{\left\| {}^{I}\overrightarrow{LOS_P} \right\| \left\| \overrightarrow{V_g} \right\|} \tag{1.10}$$

The angle η is very important to the position estimation performance of the VBTT system. When η is equal to or close to zero, the circular guidance algorithm will work correctly, and the SUAV will circle the acquired target at a constant radius. If this happens, ω - the turn rate of the LOS can be found from LOS information, $\overset{\text{ш}}{V_g}$ can be found from GPS information, and therefore ρ - the horizontal range of the target from the SUAV can be estimated. The following equation exhibits the relationship between angle η and the position estimation variable ρ.

$$\rho = \frac{\overset{\text{ш}}{V_g}}{\omega} \tag{1.11}$$

Slight variations in angle η can determine whether or not the position estimator will function correctly. As stated above, when η is equal to or near a value of zero, the circular guidance algorithm will work correctly, and ρ can be estimated. However, if η is not equal or close to zero the circular guidance algorithm will not operate correctly causing the SUAV to circle the target at an increasing or decreasing radius. When this happens, ρ cannot be estimated. The figure below demonstrates how the circular guidance algorithm will perform when η is equal to or close to zero.

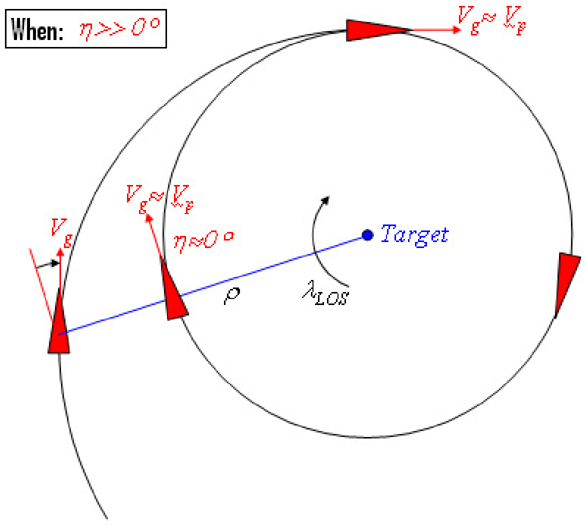

Figure 16. Circular Guidance when η is equal to or close to zero

Performance analysis of η should show its correlation with the position estimation performance. For a derivation of η and the other kinematics equations used in the circular guidance algorithm see [Ref. 1].

Data will also be taken from the Monte Carlo simulation in the form of the result matrix shown above in (1.1). The model performances that will be saved in the matrix are the convergence time of the position estimator to 50m error and 20m error from true range.

IV. RESULTS OF THE SIMULATION

A. RESULTS OF THE SIMULINK IMPLEMENTATION

1. Noise Implementation Results

When the white noise is added to the sensor, the signal itself will of course be changed. This change in signal will then change how it affects the system in which it is implemented into. This section exhibits figures that were created to show the difference in the signals between their original form and when the error is added.

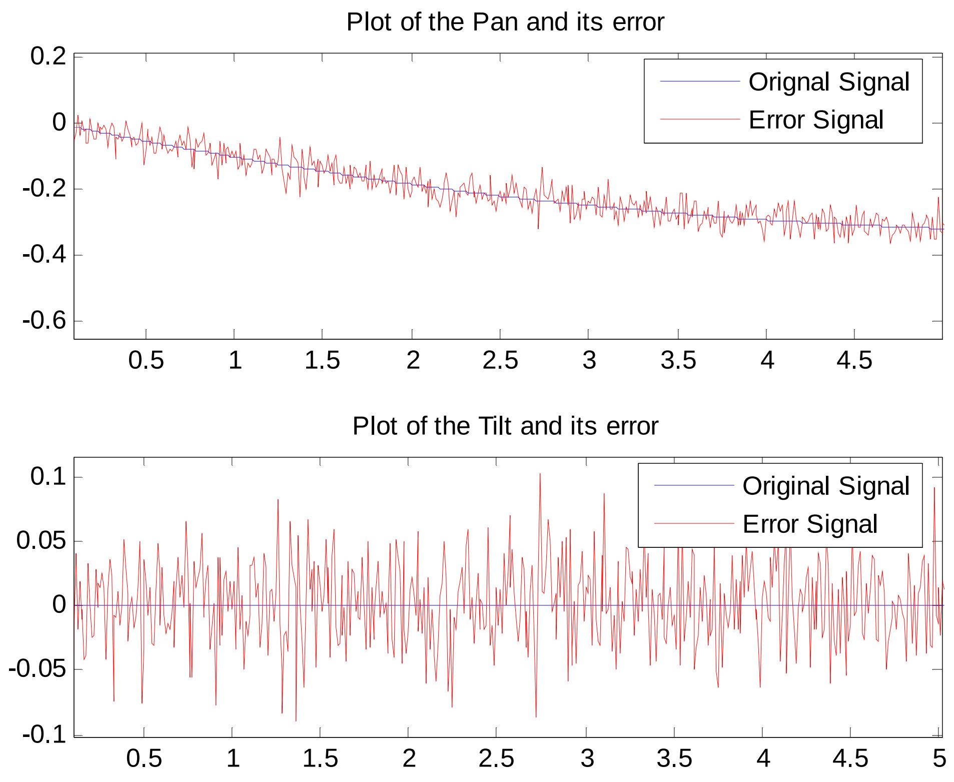

Figure 17. Plots of the Pan and Tilt Sensors with their Noise Signals

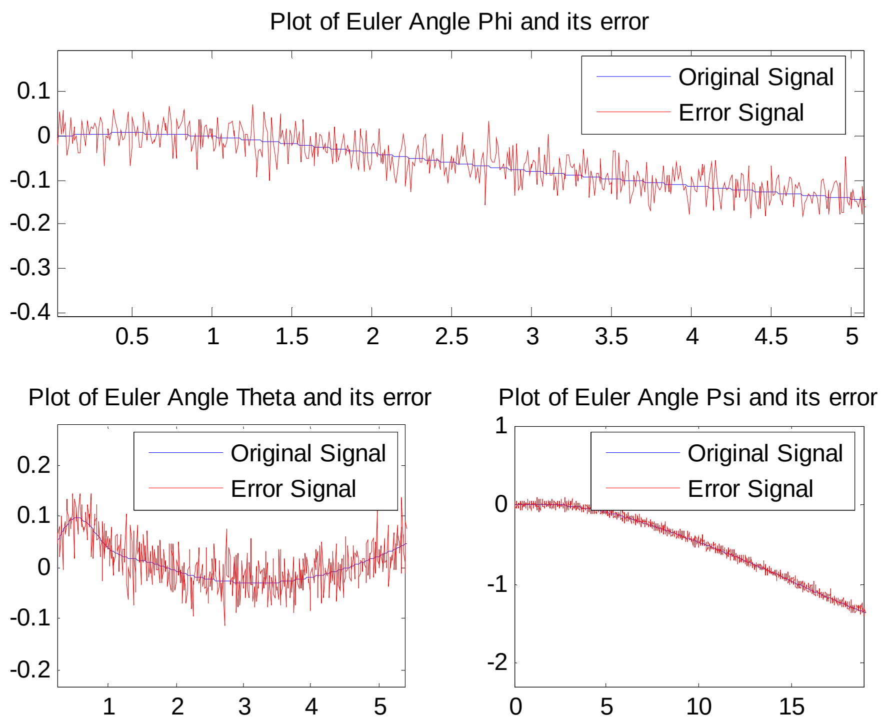

Figure 18. Plots of the Euler Angle Sensors with their Noise Signals

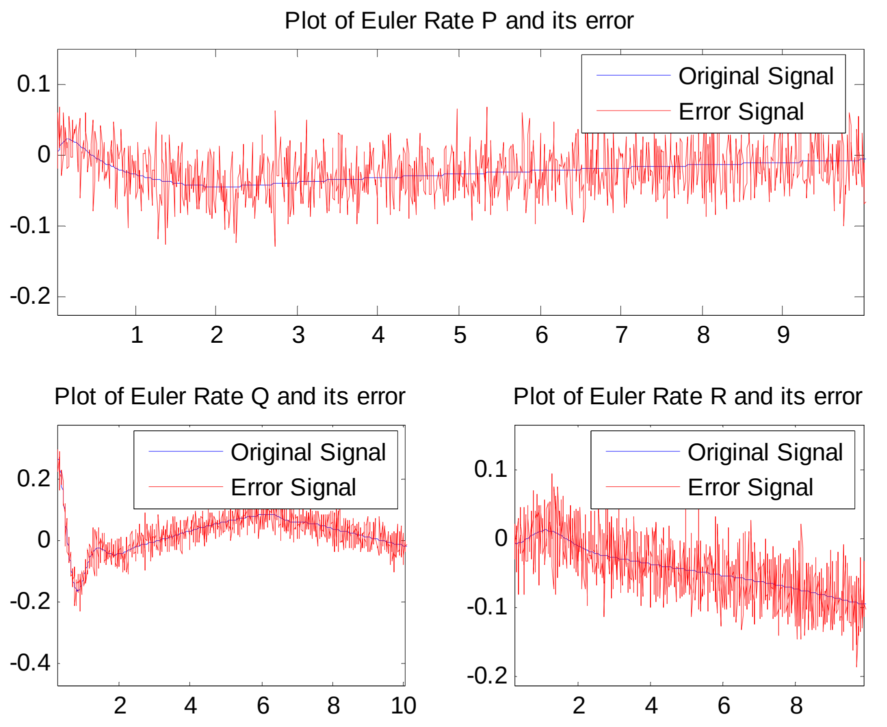

Figure 19. Plots of the Euler Rate Sensors with their Noise Signals

Figures 16, 17, and 18 displayed above show how all the sensors change with noise signals added to them. Although the plots only show the sensors with one noise iteration (0.1 mean and variance), they give a good idea of what all the error signals will actually look like. Of course in the four main scenarios of this thesis, many noise iterations will be run.

2. Ideal Simulation Results

The purpose of the ideal simulation is to acquire a data base for which all the other following simulations can and will be compared to. It is important because, after this scenario is finished running, one will be able to see how the SUAV performs under nominal circumstances. There is no external noise added to the system in the ideal model simulation.

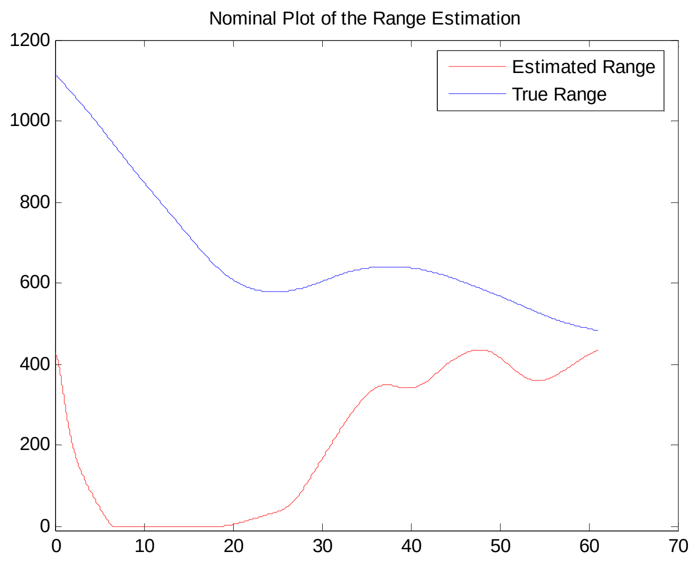

Figure 20. Range Estimation Plot for the Ideal Model Simulation

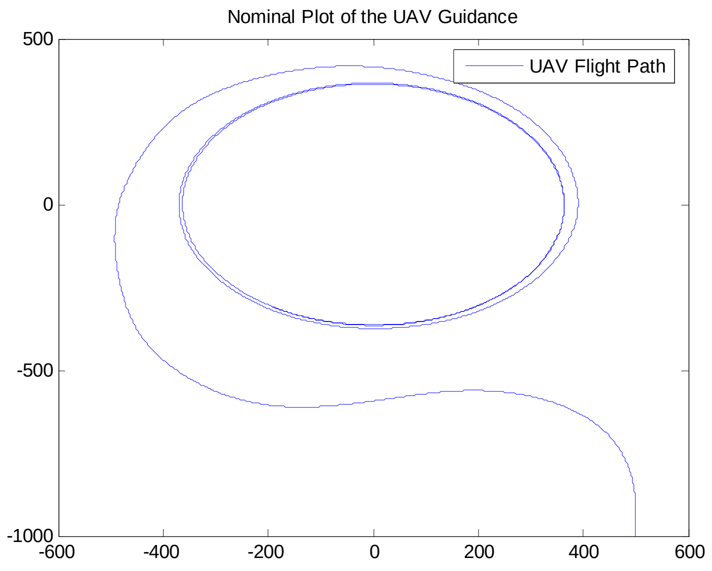

Figure 21. Guidance Plot for the Ideal Model Simulation

The two figures shown above are the results of the ideal model. Figure 19 shows how the range estimator under normal conditions. The blue track displays the true range of the target from the SUAV. The red track demonstrates how the range estimator estimates the range of the target to the SUAV. As one can easily see, the range estimation filter is off a little at first. It takes around 20 seconds in this simulation to start converging to the actual true range. This is due to angle η. The stopping criterion of the simulation was: when the estimated range converted to within 50 meters error of the true range the simulation would end. As seen in Figure 19, it takes the range estimation filter of SUAV approximately 62 seconds to converge to within 50 meters error of the target. When noise is added to the system, this convergence time will drastically increase.

Figure 20 reveals how the NPS control loop acts in an ideal situation. In the plot, the target is located at the origin (0,0), where the guidance loop tells the SUAV to circle at a fixed radius. It takes approximately 100 seconds for the UAV flight guidance to reach its destined path. In this simulation the SUAV would continue to circle around the target until it was ended or in real flight, if the NPS control law was turned off. The table below shows the matrix of results for the ideal simulation.

	Mean	Variance	Tconv 50m	Tconv 20m
Ideal	0	0	61.1834	73.05

Table 1. Result Matrix for the Ideal Simulation

3. Individual Sensor Simulation Results

The purpose of doing the individual sensor simulation is to see how each distinct sensor affects the position estimation performance of the VBTT system. This scenario was run for each of the eight chosen parameters (pan, tilt, phi ϕ, theta θ, psi ψ, *p*, *q*, and *r*) in order to acquire a sensitivity model for the position estimator. The results of this scenario are shown below.

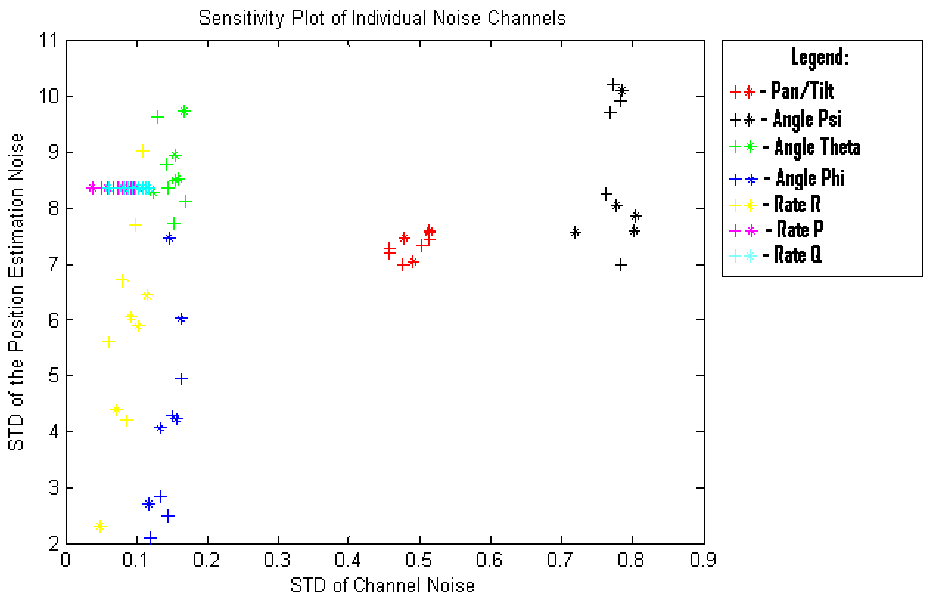

Figure 22. Sensitivity Plot for the Individual Sensor Simulation

Figure 21 includes the sensitivity plot for the individual sensor simulation. It displays the sensitivity of the position estimator to the eight individual sensors. The sensors are shown in different colors. In the plot, high sensitivity is represented by increased values of channel noise and position estimation error. Figure 21 shows that the position estimator is most sensitive to the pan and tilt angles of the gimbaled camera, and the heading angle. It can also be seen from the figure that the position estimator is least sensitive to angular rates *p* and *q*. The following tables of result matrices show the convergence times of the position estimator to 50m and 20m error. The same sensitivity trends seen in Figure 21 are shown in the tables below.

	Mean	Variance	Tconv 50m	Tconv 20m	
	0.001	0.001	69.95	83.4121	Iteration 1
	0.002	0.002	70.3137	82.7388	Iteration 2
	0.003	0.003	70.22	82.9982	Iteration 3
	0.004	0.004	69.65	82.9	Iteration 4
Pan	0.005	0.005	69.7284	83.2462	Iteration 5
	0.006	0.006	70.2403	83.2159	Iteration 6
	0.007	0.007	70.1141	83.1	Iteration 7
	0.008	0.008	70.2265	83.6582	Iteration 8
	0.009	0.009	70.321	83.452	Iteration 9
	0.01	0.01	70.2156	82.7861	Iteration 10

Table 2. Result Matrix for the Pan Sensor

Table 2 shown above displays the results for the individual Pan sensor in result matrix form. It can easily be seen that when the values of this table are compared to that of Table 1, the convergence times to fifty meters and twenty meters drastically go up. The convergence time to fifty meters goes up by an average of nine seconds, and the convergence time to twenty meters goes up by an average of about eleven seconds.

	Mean	Variance	Tconv 50m	Tconv 20m	
	0.001	0.001	61.0833	72.4382	Iteration 1
	0.002	0.002	61.0928	72.4693	Iteration 2
	0.003	0.003	61.344	72.5893	Iteration 3
	0.004	0.004	61.3084	72.5162	Iteration 4
Phi	0.005	0.005	61.0155	72.4641	Iteration 5
	0.006	0.006	61.2514	72.4476	Iteration 6
	0.007	0.007	61.5018	72.7302	Iteration 7
	0.008	0.008	61.4726	72.5448	Iteration 8
	0.009	0.009	61.3808	72.5202	Iteration 9
	0.01	0.01	61.7979	72.6674	Iteration 10

Table 3. Result Matrix for Euler Angle Phi

Table 3 shown above shows the results for the individual ϕ sensor in result matrix form. The values in Table 3 show that noise in the parameter ϕ channel has little effect on the VBTT system of the SUAV by itself.

	Mean	Variance	Tconv 50m	Tconv 20m	
	0.001	0.001	60.85	72.65	Iteration 1
	0.002	0.002	61.6374	73.016	Iteration 2
	0.003	0.003	61.05	72.75	Iteration 3
	0.004	0.004	61.263	72.6063	Iteration 4
Theta	0.005	0.005	61.1919	72.708	Iteration 5
	0.006	0.006	61.6163	72.5656	Iteration 6
	0.007	0.007	61.0129	72.7294	Iteration 7
	0.008	0.008	61.2423	72.6235	Iteration 8
	0.009	0.009	61.05	72.65	Iteration 9
	0.01	0.01	61.2686	72.6644	Iteration 10

Table 4. Result Matrix for Euler Angle Theta

The table above shows the results for sensor θ in result matrix form. It can easily be seen from the values in the table that noise in the Euler angle θ channel has minimal affect on the VBTT system of the SUAV.

	Mean	Variance	Tconv 50m	Tconv 20m	
	0.001	0.001	61.2137	72.5799	Iteration 1
	0.002	0.002	60.8399	72.5289	Iteration 2
	0.003	0.003	61.4359	72.65	Iteration 3
	0.004	0.004	61.5133	72.4917	Iteration 4
Psi	0.005	0.005	61.3133	72.5106	Iteration 5
	0.006	0.006	61.5721	72.8389	Iteration 6
	0.007	0.007	60.7778	72.5135	Iteration 7
	0.008	0.008	61.8617	72.7896	Iteration 8
	0.009	0.009	61.9239	73.0632	Iteration 9
	0.01	0.01	61.8814	73.1229	Iteration 10

Table 5. Result Matrix for Euler Angle Psi

Table 5, displayed above shows the results for the ψ sensor in result matrix form. The table clearly shows that noise in the Euler angle channel ψ has some affect on the position estimator. More on this will be discussed at the end of this section.

	Mean	Variance	Tconv 50m	Tconv 20m	
	0.001	0.001	61.1834	73.05	Iteration 1
	0.002	0.002	61.1834	73.05	Iteration 2
	0.003	0.003	61.1834	73.05	Iteration 3
	0.004	0.004	61.1834	73.05	Iteration 4
P	0.005	0.005	61.1834	73.05	Iteration 5
	0.006	0.006	61.1834	73.05	Iteration 6
	0.007	0.007	61.1834	73.05	Iteration 7
	0.008	0.008	61.1834	73.05	Iteration 8
	0.009	0.009	61.1834	73.05	Iteration 9
	0.01	0.01	61.1834	73.05	Iteration 10

Table 6. Result Matrix for Euler Rate P

Table 6 shown above displays the results for the individual sensor *p* in result matrix form. When the values of this table are compared to that of Table 1, a trend can be spotted. The values are exactly the same. It can therefore be concluded that noise in the Euler rate *p* channel has no affect on the VBTT system of the SUAV.

	Mean	Variance	Tconv 50m	Tconv 20m	
	0.001	0.001	61.1834	73.05	Iteration 1
	0.002	0.002	61.1834	73.05	Iteration 2
	0.003	0.003	61.1834	73.05	Iteration 3
	0.004	0.004	61.1834	73.05	Iteration 4
Q	0.005	0.005	61.1834	73.05	Iteration 5
	0.006	0.006	61.1834	73.05	Iteration 6
	0.007	0.007	61.1834	73.05	Iteration 7
	0.008	0.008	61.1834	73.05	Iteration 8
	0.009	0.009	61.1834	73.05	Iteration 9
	0.01	0.01	61.1834	73.05	Iteration 10

Table 7. Result Matrix for Euler Rate Q

The table above exhibits the results for sensor *q* in result matrix form. Just like the Euler rate *p*, *q* shares the same values with the ideal simulation. Clearly it has no affect on the position estimator of the SAUV.

	Mean	Variance	Tconv 50m	Tconv 20m	
	0.001	0.001	61.1835	72.5298	Iteration 1
	0.002	0.002	61	72.4836	Iteration 2
	0.003	0.003	60.9899	72.5329	Iteration 3
	0.004	0.004	61.268	72.6291	Iteration 4
R	0.005	0.005	61.0902	72.55	Iteration 5
	0.006	0.006	61.2343	72.85	Iteration 6
	0.007	0.007	60.9557	72.65	Iteration 7
	0.008	0.008	61.0326	72.7265	Iteration 8
	0.009	0.009	61.3124	73.0351	Iteration 9
	0.01	0.01	61.5126	73.2	Iteration 10

Table 8. Result Matrix for Euler Rate R

Table 8, shown above shows the results for the *r* sensor in result matrix form. The table clearly shows that noise in the Euler rate channel *r* has some affect on the position estimator. The influence over the position estimator for Euler rate *r* is very small, however it can be seen that there is at least a minimal affect on the VBTT system.

The results of the individual sensor simulations clearly show the sensitivity of the VBTT system. Figure 21 and Table 2 clearly demonstrate that the pan of the gimbaled camera has the most affect on the position estimator. Therefore, it can be said that the VBTT system is most sensitive to the pan of the gimbaled camera. The tables also show that noise in the Euler angle sensors have small affect on the system. The results of this simulation have also proven that the VBTT system is not sensitive at all to the Euler rates *p* and *q*. This can be seen quite easily from the tables or from Figure 21, where these two Euler rates are displayed as horizontal lines close to the left side of the plot.

4. Target-Tracking Simulation Results

As stated in the sections above, the purpose of the target-tracking scenario is to demonstrate how the state of target-tracking affects the ability of the range estimator. This simulation included loss of track events: 25% of the time, 55% of the time, and finally 85% of the time. The results for these three parts of the scenario are shown below.

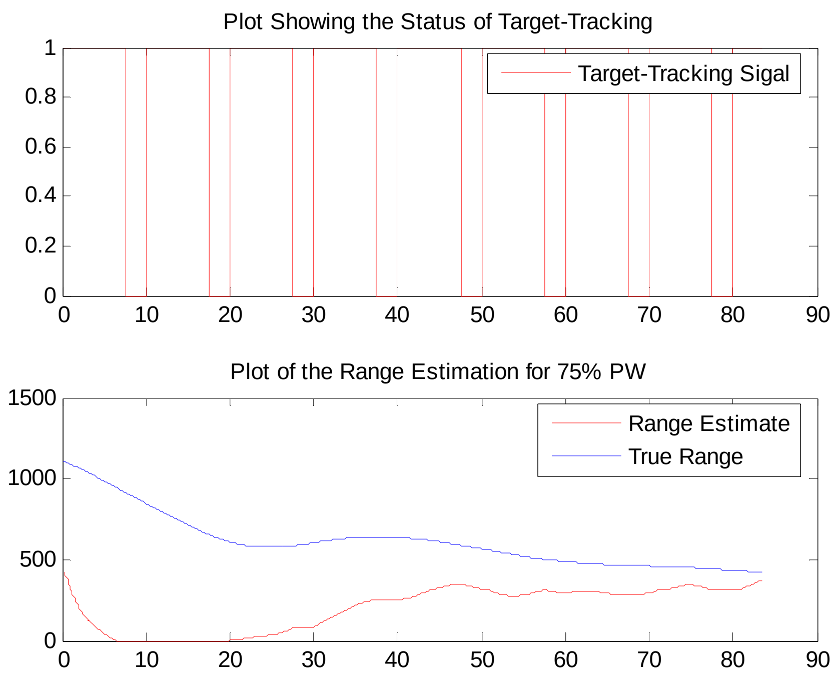

Figure 23. Plot of the Range Estimation for 25% Target-Tracking Error

Figure 22 shown above displays how the range estimation filter behaves when track is lost 25% of the time. This means that the target is only being tracked three quarters of the time, simulating target loss and then target acquisition. The first subplot of Figure 22 shows just this. In this subplot, the red signal is at a value of 1(target captured) for 7.5 out of a 10 second period. For the remaining 2.5 seconds, the target is lost and a value of 0 is displayed. The second subplot shows the range estimation ability of the filter at this iteration. Just like in Figure 19, the ideal range estimation plot, the filter takes about 20 seconds to start converging to the true range of the target. However this time the filter takes approximately 85 seconds to converge within 50 meters error of the true range. This plot clearly shows how sensitive the filter and the system are to just 25% target-tracking error. This convergence time will surely increase as the percent error of the target-tracking capability is increased.

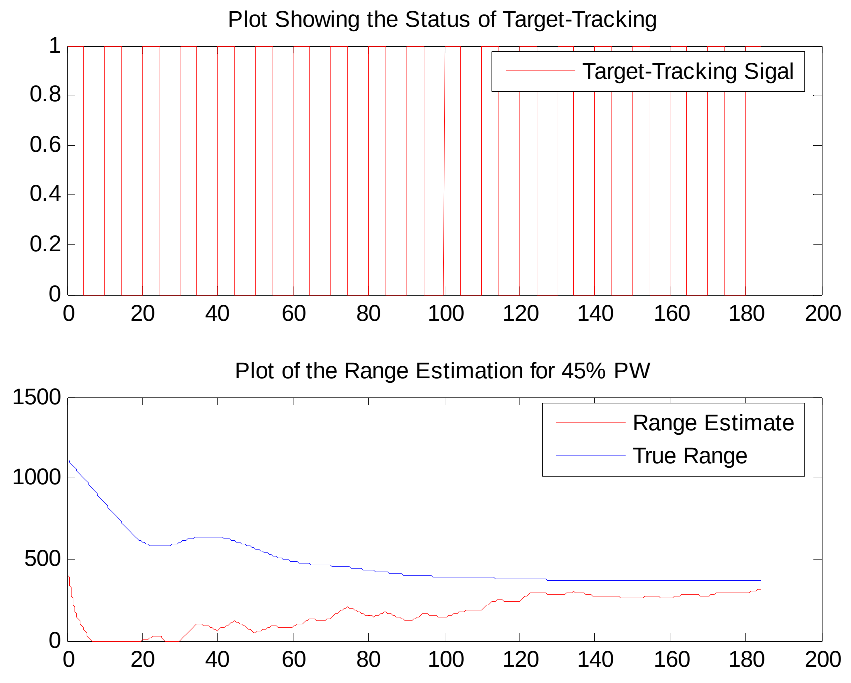

Figure 24. Plot of the Range Estimation for 55% Target-Tracking Error

Figure 23 shows basically the same thing as Figure 22, except this time the target is lost 55% of the time. Again, the first subplot displays the target-tracking signal, and how for this scenario the SUAV is only locked onto the target 45% of the time. The second subplot shows how the range estimator acted for this iteration. Just like in the ideal plot, it took the filter around 20 seconds to start converging to the true range. However, unlike the nominal plot and Figure 21, it takes the filter a lot longer to converge to within 50 meters of the true range. It takes approximately 185 seconds to converge to 50 meters of the true range. This is over two times as long as it took for the previous iteration. This figure shows that the sensitivity of the system goes up significantly if the target track loss is increased to anything over 25% of the time.

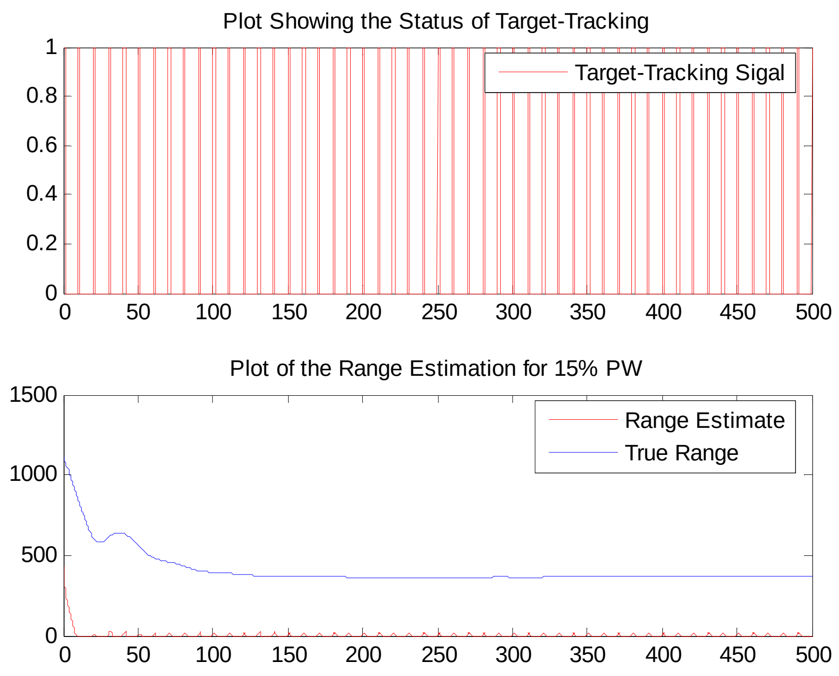

Figure 25. Plot of the Range Estimation for 85% Target-Tracking Error

Figure 24 shows the final iteration performed in this simulation. For this iteration, target track is lost 85% of the time. It reveals a lot about the sensitivity of the range estimation filter to the error in target-tracking. Clearly the estimator does not work with target loss duration of this magnitude.

Input	Tconv 50m	Tconv 20m	
0.75	83.5792	84.9969	Iteration 1
0.45	294.35	384.5759	Iteration 2
0.15	inf	inf	Iteration 3

Table 9. Result Matrix for the Target-Tracking Simulation

This simulation has clearly shown how sensitive the range estimation is to target loss events. The PerceptiVU target tracking software is a critical part of the NPS SUAV and how it carries out its mission. Simulation results exhibit how important it is for the SUAV project team to keep the PerceptiVU software up to date and working with minimal errors.

5. Final Simulation Results

The purpose of this simulation is to determine the performance of the VBTT system in the presence of errors in all channels. By doing this an overall sensitivity model for the guidance angle η was achieved. This sensitivity plot for η is shown below.

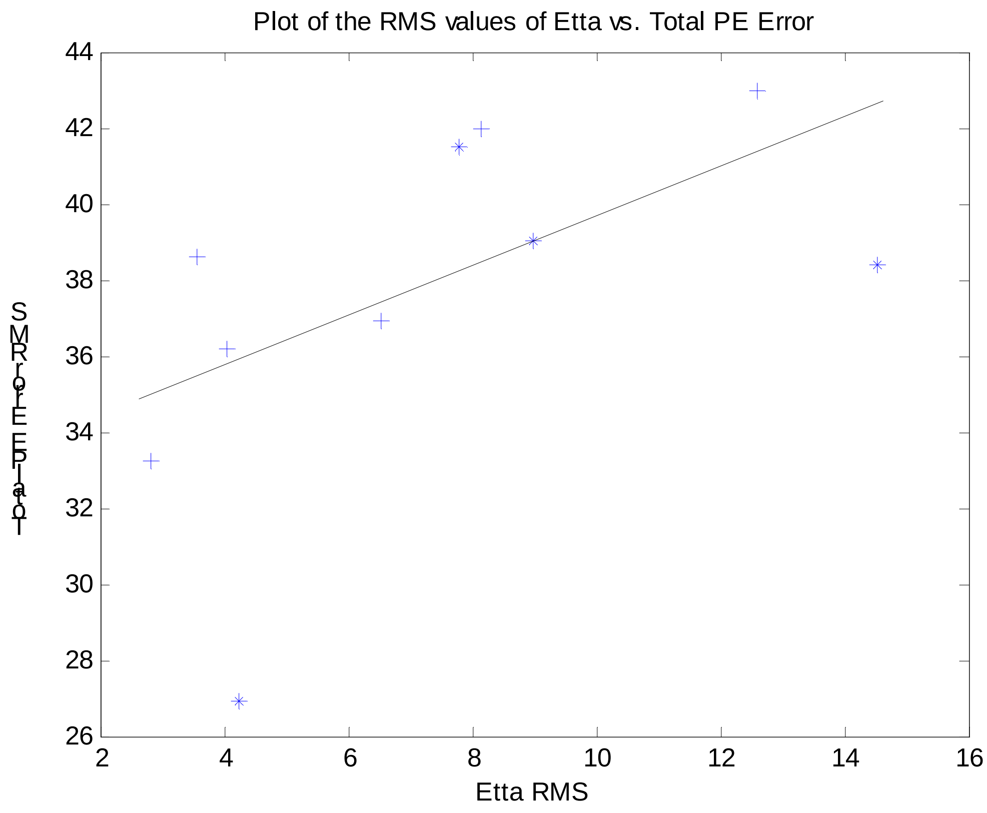

Figure 26. Plot of Position Estimator Angle η

Figure 25 shown above demonstrates the sensitivity of η to noise in all channels of the system. The plot exhibits a trend in η as increased amounts of noise is implemented into the system. The blue stars and crosses in the plot represent RMS values of η with respect to RMS values of the error in the position estimator. It can be easily seen from the figure that as increased amounts of noise is added into the system the RMS values of η will increase. A best-fit line was inserted into the figure so that the trend exhibited by η could be easily seen. Of course this full noise scenario was only run for a total of ten noise iterations. If it were run for one-thousand iterations the trend in η

would be the same, it would linearly increase with an increasing amount of noise. The findings of this simulation are also shown below in result matrix form.

Mean	Variance	Tconv 50m	Tconv 20m	
0.001	0.001	69.8805	83.1443	Iteration 1
0.002	0.002	70.1722	83.6308	Iteration 2
0.003	0.003	70.7286	96.0886	Iteration 3
0.004	0.004	71.0922	95.8413	Iteration 4
0.005	0.005	71.3536	83.8265	Iteration 5
0.006	0.006	71.0975	84.5742	Iteration 6
0.007	0.007	79.65	inf	Iteration 7
0.008	0.008	72.9163	98.1141	Iteration 8
0.009	0.009	79.0882	96.2	Iteration 9
0.01	0.01	74.1126	95.5459	Iteration 10

Table 10. Result Matrix for the Full Noise Simulation

Table 10, shown above displays the results of the full noise simulation in result matrix form. The results in this table prove to be very promising for the SUAV project. It is seen that with increased noise in all channels of the system that the values for convergence time go up drastically but do not blow up. This characteristic is very important to the performance of the VBTT system. The table shows that the SUAV exhibits graceful degradation of performance.

V. CONCLUSIONS

Completion of this thesis has shown the NPS SUAV team many important details concerning performance of the SUAV and its VBTT system. The ideal scenario demonstrated how the SUAV performed under normal circumstances with no noise involved. It was shown that in full scale nonlinear simulation the position estimation capability of the SUAV takes roughly 62 seconds to converge to a reasonable range.

The three Monte Carlo scenarios run in this thesis have also been insightful to the NPS SUAV project team. The simulation of individual channel noise gave an overall sensitivity model for the VBTT system. It exhibited that the VBTT system is most sensitive to noise in the pan and tilt of the gimbaled camera, as well as noise in Euler angle ψ. Simulation of the target-tracking scenario proved that the VBTT system is overly sensitive to increased target loss. Completion of the full noise scenario showed that the guidance angle η becomes increasingly sensitive, negatively affecting the VBTT system performance.

Results of both the individual sensor and full noise simulations confirm that improvements can be made to the gimbal of the SUAV. This includes the physical gimbal actuation and calibration, and the gimbal control model. Since the VBTT system is so sensitive to pan and tilt errors in the gimbal, it is of the utmost importance for the NPS SUAV team to keep the gimbal hardware and software, including PerceptiVU updated and error free. The intuitive result of the target-tracking scenario can greatly help the NPS SUAV team improve upon the VBTT system. Through this outcome, it is apparent that the VBTT system must be modified so that the position estimation capability can converge at a minimal time. Work to improve and ultimately solve this problem is currently underway. Through manipulation of certain Kalman filter gains inside the SUAV model, the team can possibly decrease the convergence time of the position estimator significantly. Careful examination of this subject could prove to be the next big milestone of the NPS SUAV project.

With the increasing movement of overly expensive UAV being used to carry out vision-based target-tracking missions, the NPS SUAV could be the beginning of a new trend. The NPS SUAV provides for all the same mission needs as its more expensive counterparts at a fraction of the cost. It is of the utmost importance for the NPS SUAV team to continue pushing the limits of UAV technology and performance so that the armed forces can benefit. The Naval Postgraduate School, with its mixture of professors with years of industry experience and military aviators, provides a perfect environment for such research.

APPENDIX A: SETTING UP THE SIMULATIONS

The Simulink simulation was set up and executed in a series of four steps for each of the scenarios. These steps include introducing the noise, running the simulation, saving the simulation, and finally obtaining the sensitivity plot or results matrix. These steps will be discussed in detail in the following sections of this appendix.

A. INTRODUCING THE NOISE

The first step for implementing the noise to the Simulink model shown above was to introduce white noise into the system. The easiest way to accomplish this task was to go into the models subsystems and find where the signals for pan, tilt, Euler angles, and Euler rates were located, and add a random number generator to them. The addition of the noise to the signal creates a new one which is introduced into the SUAV control system. The creation of this new noise signal was done inside of subsystem blocks, so that things would not look so cluttered in the model. Manual switches were also incorporated into the model after the random noise generating subsystems, so that the subsystems could be cut off from the control system if the nominal model had to be run.

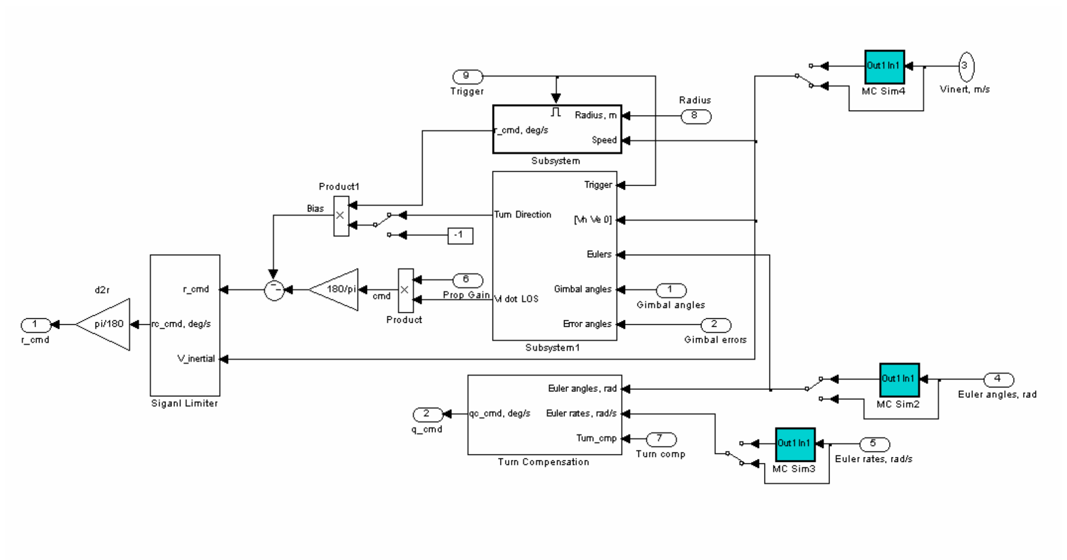

Figure 27. Simulink Model, Sim3/UAV 6DOF simulation/CurGuid Controller

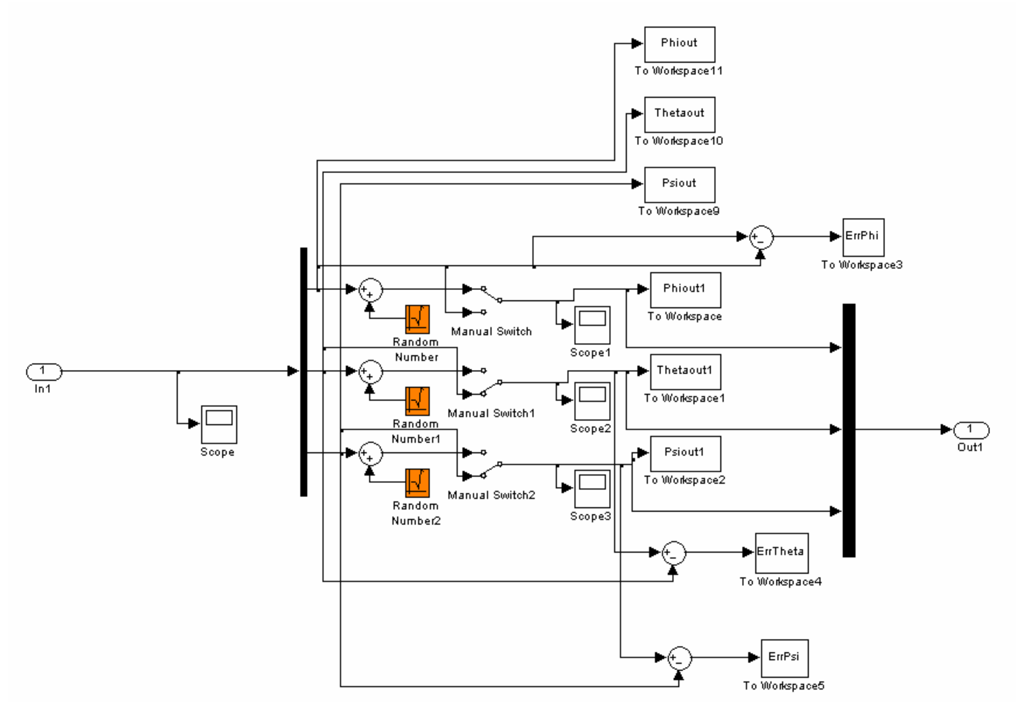

Figure 28. White Noise Generating Subsystem, Sim3/UAV 6DOF simulation/CurGuid Controller/MC Sim2

The two figures above show a partial representation of how the white noise is implemented into the system. Figure 26, shown above, displays the noise generating subsystems for the Euler angles, and Euler rates. These noise generating subsystems are shown in light blue. The subsystem that produces the white noise for the pan and tilt of the gimbaled camera is not shown for the interest of conserving space, but it can be found in the model Sim3/UAV 6DOF simulation/Gimballed Camera Model. Figure 27, shown above, exhibits what is inside of the subsystem named MC Sim2. In this figure, the Euler angles signal from the in port is first demuxed into its three parts ϕ, θ, and ψ. Once this is done, random number producers (shown as orange block in Figure 27) are added to each of the three sensors, creating new signals for each. ϕ, θ, and ψ are then muxed back together into one signal again, and sent to the out port. Once the signal reaches the out port of the subsystem, it is sent to the rest of the control system. This same idea was used when implementing the noise for all of the other sensors given above, but cannot be shown in the interest of preserving space.

When a random number generating block is placed into a Simulink model, the random signal is given starting values for its mean and variance. In this thesis, the values of the signals mean and variance will be changing in iterations. These iteration values were discussed above as values X_i of the results matrix. How the noise iteration X_i will be adjusted for the purpose of the simulations will be discussed in the next section entitled running the simulation.

Implementing white noise to the last of the nine sensors, pulse width, had to be done a little differently. The purpose of pulse generator in Sim3 is to simulate a pulse telling when the target-tracking system is locked onto a target, and when it is not locked on to a target. When the pulse generator is giving off a value of one, the VBTT system is currently locked onto a target. When the pulse generator is giving off a value of zero, the VBTT system is currently seeking to acquire a target. This is important because by using the pulse generator we can manipulate when the VBTT system is tracking and when it is not. This will show us how error influence over the VBTT system will affect model performance of the range estimator. The pulse generator that will be implemented for the target-tracking simulation, and can be seen in Figure 14, located above. In Figure 14, it can also be seen that a manual switch is located after the pulse generator so that this feature can be taken out of the system if needed.

B. RUNNING THE SIMULATIONS

The procedure for running the four main simulations is the same. First, the user must flip the proper manual switches on or off, depending which scenario is to be run. If the ideal scenario is going to be run, all manual switches must be flipped off so that no external noise reaches the system, and vice versa for when all noise channels are to be turned on.

After the manual switches have been placed in the correct position, the needed error iterations must be implemented to the random number generators. This can be done manually by finding the random number generating blocks in their subsystems and changing their mean and variance values. Setting the parameters in this fashion, although effective is very time consuming, and can be done automatically. The automatic procedure for setting parameters was chosen to speed along the simulation process. The ‘set_param’ Matlab command was used to set parameters of a given block inside a Simulink model from the command line or an m-file. Once the set_param command has been used to set the needed parameters in the Simulink model, the model could be run. This could also be done in many different ways, the most obvious being to press the play button in the top menu of the Simulink model. This however, can be taxing because if you want to run the model for a fixed amount of time, you must watch the timer and press the stop button. This problem is fixed by using the ‘sim’ Matlab command from the command line or an m-file. With this simple command, one can easily run a Simulink model for a given amount of time, and it will stop the simulation at the prescribed time. As you can see, Matlab commands are primarily used to run the simulation. It was found that using these commands plus the commands in the following two sections in an m-file made the task of simulating a model many times a very easy process.

C. SAVING THE SIMULATIONS

The procedure for saving scenario results was also made very easy to accomplish by using a Matlab command in an m-file. Once the given scenario has been run, certain parameters like the time, mean and variance iteration, true range, and estimated range needed to be saved so that they could later be analyzed from the workspace to help build the sensitivity models. This was easily done through use of the ‘save’ command in Matlab. The save command will save any specified model parameter to a .mat file as long as that parameter appears in the Matlab workspace after the model has been run. This .mat file can then be opened at a later time, and the saved parameters can be called from the Matlab workspace for analysis.

D. ATTAINING SIMULATION RESULTS

After the simulations were run, and the needed .mat files were created and saved, the results of the simulations could be obtained. The results of the simulations came in many forms. The first is the data put into the result matrix M as described in above sections. The other way that the results will be presented is in plot form. Many plots were be created through use of m-files calling data from .mat files. These many plots include figures of the SUAV guidance, comparison of true range and estimated range vs. time, and sensitivity model plots comparing the influence of the nine parameters on the position estimation performance. In order to make the plotted results for the sensitivity models more presentable, standard deviations and RMS values were calculated and plotted.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX B: MATLAB ROUTINES

A. IDEAL ROUTINE

```
%=================================================================
%%%%%Ideal Scenario
%%%%%by ENS Todd Trago
%=================================================================
%%%%This m-file only deals with none of the parameters
%%%%from Sim2.mdl or Sim3.mdl.
%%%The file Sim2.mdl must be opened and in the current
%%%directory for this to work correctly
%=================================================================
%-------------------------------------------------------------------
%%Run the Simulink model with no noise put into the system, so the nominal
%%values can be acquired.  In order to do this, the user must first go into
%%the model in the correct subsystems and make sure that all the manual
%%switches are turned off, so that no noise is added into the system
%Run the nominal simulation
[t]=sim('Sim2',[0 100]);
%Name the parameters that are needed to be saved.
T_RangeN=T_Range;
oneN=one;
twoN=two;
%Save Simulation results in a .mat file
%Savefile= 'Nominal.mat';
save('Nominal','t','T_RangeN','oneN','twoN');
%Get the Range estimation plot
figure(1)
plot(t,oneN,'r',t,T_RangeN,'b');legend('oneN','True RangeN','Location','Northeast');
title('Plot of the Range Estimation Nominal');
hold on
%Guidance plot
figure(2)
plot(XYN(:,1),XYN(:,2));legend('UAV Flight Path','Location','Northeast');
title('Nominal Plot of the UAV Guidance');
hold on
clear all
%-------------------------------------------------------------------
%=================================================================
```

B. INDIVIDUAL SENSOR ROUTINES

1. Individual Pan/Tilt Simulation M-File

```
 %=============================================================
%%%%%Idividual Scenario of Pan noise
%%%%%by ENS Todd Trago
%=============================================================
%=============================================================
%-------------------------------------------------------------------
%%This script file will implement noise to the parameter Pan
%%to see how it affects model performances.  This will be performed 10
%%times so that an overall plot of the Pan performance can be attained.
%Pan**************************************************************
```

```
%-------------------------------------------------------------------
%Set the paramters for the first Pan simulation
set_param('Sim3/UAV 6DOF simulation/Gimballed Camera model/MC Sim1/Random
Number2','Variance','0.001','Mean','0.001');
%Run the simulation
[t]=sim('Sim3',[0 30]);
%Name the parameters that are needed to be saved.
T_RangeP1=T_Range;
oneP1=one;
twoP1=two;
ConvP1=Conv;
%Find the RMS Values of PE convergence and the Pan/Tilt angles
y1= sqrt(sum((ConvP1).*conj(ConvP1))/size((ConvP1),1));
y2= sqrt(sum((Panangles1).*conj(Panangles1))/size((Panangles1),1));
%Acquire the standard deviations and convariances of the position
%estimation convergence and the pan.
StandPE1=std(ConvP1);
CovPE1=cov(ConvP1);
StandPan1=std(Panangles1);
CovPan1=cov(Panangles1);
%Save Simulation results in a .mat file
%Savefile= 'Pan1.mat';
save('Pan1','t','T_RangeP1','oneP1','twoP1','ConvP1','Panangles1','StandPE1','CovPE1','StandPan1'
,'CovPan1','y1','y2');
clear all
%-------------------------------------------------------------------
%Set the paramters for the second Pan simulation
set_param('Sim3/UAV 6DOF simulation/Gimballed Camera model/MC Sim1/Random
Number2','Variance','0.002','Mean','0.002');
%Run the simulation
[t]=sim('Sim3',[0 30]);
%Name the parameters that are needed to be saved.
T_RangeP2=T_Range;
oneP2=one;
twoP2=two;
ConvP2=Conv;
Panangles2=Panangles1;
%Find the RMS Values of PE convergence and the Pan/Tilt angles
y3= sqrt(sum((ConvP2).*conj(ConvP2))/size((ConvP2),1));
y4= sqrt(sum((Panangles2).*conj(Panangles2))/size((Panangles2),1));
%Acquire the standard deviations and convariances of the position
%estimation convergence and the pan.
StandPE2=std(ConvP2);
CovPE2=cov(ConvP2);
StandPan2=std(Panangles2);
CovPan2=cov(Panangles2);
%Save Simulation results in a .mat file
%Savefile= 'Pan2.mat';
save('Pan2','t','T_RangeP2','oneP2','twoP2','ConvP2','Panangles2','StandPE2','CovPE2','StandPan2'
,'CovPan2','y3','y4');
clear all
%-------------------------------------------------------------------
%Set the paramters for the Third Pan simulation
set_param('Sim3/UAV 6DOF simulation/Gimballed Camera model/MC Sim1/Random
Number2','Variance','0.003','Mean','0.003');
%Run the simulation
```

```
[t]=sim('Sim3',[0 30]);
%Name the parameters that are needed to be saved.
T_RangeP3=T_Range;
oneP3=one;
twoP3=two;
ConvP3=Conv;
Panangles3=Panangles1;
%Find the RMS Values of PE convergence and the Pan/Tilt angles
y5= sqrt(sum((ConvP3).*conj(ConvP3))/size((ConvP3),1));
y6= sqrt(sum((Panangles3).*conj(Panangles3))/size((Panangles3),1));
%Acquire the standard deviations and convariances of the position
%estimation convergence and the pan.
StandPE3=std(ConvP3);
CovPE3=cov(ConvP3);
StandPan3=std(Panangles3);
CovPan3=cov(Panangles3);
%Save Simulation results in a .mat file
%Savefile= 'Pan3.mat';
save('Pan3','t','T_RangeP3','oneP3','twoP3','ConvP3','Panangles3','StandPE3','CovPE3','StandPan3'
,'CovPan3','y5','y6');
clear all
%------------------------------------------------------------------
%Set the paramters for the fourth Pan simulation
set_param('Sim3/UAV 6DOF simulation/Gimballed Camera model/MC Sim1/Random
Number2','Variance','0.004','Mean','0.004');
%Run the simulation
[t]=sim('Sim3',[0 30]);
%Name the parameters that are needed to be saved.
T_RangeP4=T_Range;
oneP4=one;
twoP4=two;
ConvP4=Conv;
Panangles4=Panangles1;
%Find the RMS Values of PE convergence and the Pan/Tilt angles
y7= sqrt(sum((ConvP4).*conj(ConvP4))/size((ConvP4),1));
y8= sqrt(sum((Panangles4).*conj(Panangles4))/size((Panangles4),1));
%Acquire the standard deviations and convariances of the position
%estimation convergence and the pan.
StandPE4=std(ConvP4);
CovPE4=cov(ConvP4);
StandPan4=std(Panangles4);
CovPan4=cov(Panangles4);
%Save Simulation results in a .mat file
%Savefile= 'Pan4.mat';
save('Pan4','t','T_RangeP4','oneP4','twoP4','ConvP4','Panangles4','StandPE4','CovPE4','StandPan4'
,'CovPan4','y7','y8');
clear all
%------------------------------------------------------------------
%Set the paramters for the fifth Pan simulation
set_param('Sim3/UAV 6DOF simulation/Gimballed Camera model/MC Sim1/Random
Number2','Variance','0.005','Mean','0.005');
%Run the simulation
[t]=sim('Sim3',[0 30]);
%Name the parameters that are needed to be saved.
T_RangeP5=T_Range;
oneP5=one;
```

```
twoP5=two;
ConvP5=Conv;
Panangles5=Panangles1;
%Find the RMS Values of PE convergence and the Pan/Tilt angles
y9= sqrt(sum((ConvP5).*conj(ConvP5))/size((ConvP5),1));
y10= sqrt(sum((Panangles5).*conj(Panangles5))/size((Panangles5),1));
%Acquire the standard deviations and convariances of the position
%estimation convergence and the pan.
StandPE5=std(ConvP5);
CovPE5=cov(ConvP5);
StandPan5=std(Panangles5);
CovPan5=cov(Panangles5);
%Save Simulation results in a .mat file
%Savefile= 'Pan5.mat';
save('Pan5','t','T_RangeP5','oneP5','twoP5','ConvP5','Panangles5','StandPE5','CovPE5','StandPan5'
,'CovPan5','y9','y10');
clear all
%-------------------------------------------------------------------
%Set the paramters for the sixth Pan simulation
set_param('Sim3/UAV 6DOF simulation/Gimballed Camera model/MC Sim1/Random
Number2','Variance','0.006','Mean','0.006');
%Run the simulation
[t]=sim('Sim3',[0 30]);
%Name the parameters that are needed to be saved.
T_RangeP6=T_Range;
oneP6=one;
twoP6=two;
ConvP6=Conv;
Panangles6=Panangles1;
%Find the RMS Values of PE convergence and the Pan/Tilt angles
y11= sqrt(sum((ConvP6).*conj(ConvP6))/size((ConvP6),1));
y12= sqrt(sum((Panangles6).*conj(Panangles6))/size((Panangles6),1));
%Acquire the standard deviations and convariances of the position
%estimation convergence and the pan.
StandPE6=std(ConvP6);
CovPE6=cov(ConvP6);
StandPan6=std(Panangles6);
CovPan6=cov(Panangles6);
%Save Simulation results in a .mat file
%Savefile= 'Pan6.mat';
save('Pan6','t','T_RangeP6','oneP6','twoP6','ConvP6','Panangles6','StandPE6','CovPE6','StandPan6'
,'CovPan6','y11','y12');
clear all
%-------------------------------------------------------------------
%Set the paramters for the seventh Pan simulation
set_param('Sim3/UAV 6DOF simulation/Gimballed Camera model/MC Sim1/Random
Number2','Variance','0.007','Mean','0.007');
%Run the simulation
[t]=sim('Sim3',[0 30]);
%Name the parameters that are needed to be saved.
T_RangeP7=T_Range;
oneP7=one;
twoP7=two;
ConvP7=Conv;
Panangles7=Panangles1;
%Find the RMS Values of PE convergence and the Pan/Tilt angles
```

```
y13= sqrt(sum((ConvP7).*conj(ConvP7))/size((ConvP7),1));
y14= sqrt(sum((Panangles7).*conj(Panangles7))/size((Panangles7),1));
%Acquire the standard deviations and convariances of the position
%estimation convergence and the pan.
StandPE7=std(ConvP7);
CovPE7=cov(ConvP7);
StandPan7=std(Panangles7);
CovPan7=cov(Panangles7);
%Save Simulation results in a .mat file
%Savefile= 'Pan7.mat';
save('Pan7','t','T_RangeP7','oneP7','twoP7','ConvP7','Panangles7','StandPE7','CovPE7','StandPan7'
,'CovPan7','y13','y14');
clear all
%------------------------------------------------------------------
%Set the paramters for the eigth Pan simulation
set_param('Sim3/UAV 6DOF simulation/Gimballed Camera model/MC Sim1/Random
Number2','Variance','0.008','Mean','0.008');
%Run the simulation
[t]=sim('Sim3',[0 30]);
%Name the parameters that are needed to be saved.
T_RangeP8=T_Range;
oneP8=one;
twoP8=two;
ConvP8=Conv;
Panangles8=Panangles1;
%Find the RMS Values of PE convergence and the Pan/Tilt angles
y15= sqrt(sum((ConvP8).*conj(ConvP8))/size((ConvP8),1));
y16= sqrt(sum((Panangles8).*conj(Panangles8))/size((Panangles8),1));
%Acquire the standard deviations and convariances of the position
%estimation convergence and the pan.
StandPE8=std(ConvP8);
CovPE8=cov(ConvP8);
StandPan8=std(Panangles8);
CovPan8=cov(Panangles8);
%Save Simulation results in a .mat file
%Savefile= 'Pan8.mat';
save('Pan8','t','T_RangeP8','oneP8','twoP8','ConvP8','Panangles8','StandPE8','CovPE8','StandPan8'
,'CovPan8','y15','y16');
clear all
%------------------------------------------------------------------
%Set the paramters for the ninth Pan simulation
set_param('Sim3/UAV 6DOF simulation/Gimballed Camera model/MC Sim1/Random
Number2','Variance','0.009','Mean','0.009');
%Run the nominal simulation
[t]=sim('Sim3',[0 30]);
%Name the parameters that are needed to be saved.
T_RangeP9=T_Range;
oneP9=one;
twoP9=two;
ConvP9=Conv;
Panangles9=Panangles1;
%Find the RMS Values of PE convergence and the Pan/Tilt angles
y17= sqrt(sum((ConvP9).*conj(ConvP9))/size((ConvP9),1));
y18= sqrt(sum((Panangles9).*conj(Panangles9))/size((Panangles9),1));
%Acquire the standard deviations and convariances of the position
%estimation convergence and the pan.
```

```
StandPE9=std(ConvP9);
CovPE9=cov(ConvP9);
StandPan9=std(Panangles9);
CovPan9=cov(Panangles9);
%Save Simulation results in a .mat file
%Savefile= 'Pan9.mat';
save('Pan9','t','T_RangeP9','oneP9','twoP9','ConvP9','Panangles9','StandPE9','CovPE9','StandPan9'
,'CovPan9','y17','y18');
clear all
%-------------------------------------------------------------------
%Set the paramters for the tenth Pan simulation
set_param('Sim3/UAV 6DOF simulation/Gimballed Camera model/MC Sim1/Random
Number2','Variance','0.01','Mean','0.01');
%Run the simulation
[t]=sim('Sim3',[0 30]);
%Name the parameters that are needed to be saved.
T_RangeP10=T_Range;
oneP10=one;
twoP10=two;
ConvP10=Conv;
Panangles10=Panangles1;
%Find the RMS Values of PE convergence and the Pan/Tilt angles
y19= sqrt(sum((ConvP10).*conj(ConvP10))/size((ConvP10),1));
y20= sqrt(sum((Panangles10).*conj(Panangles10))/size((Panangles10),1));
%Acquire the standard deviations and convariances of the position
%estimation convergence and the pan.
StandPE10=std(ConvP10);
CovPE10=cov(ConvP10);
StandPan10=std(Panangles10);
CovPan10=cov(Panangles10);
%Save Simulation results in a .mat file
%Savefile= 'Pan10.mat';
save('Pan10','t','T_RangeP10','oneP10','twoP10','ConvP10','Panangles10','StandPE10','CovPE10','S
tandPan10','CovPan10','y19','y20');
clear all
%-------------------------------------------------------------------
%==============================================================
```

Note: The m-files for the other seven parameters are not shown. They do however follow the same format as the script file shown above.

2. Sensitivity Plot M-file

```
%==============================================================
%%%%%Final Sensitivity Plots for Standard Deviations
%%%%%by ENS Todd Trago
%==============================================================
%%This script file produces the final plots for all the parameters verus each other.  It can
%%only be run after the file Pan_Tilt.m has be successfully run.  This script
%%file creates one plots for the standard deviations of all the parameters
%%vs. the PE
%-------------------------------------------------------------------
%Plot the Standard Deviations of the Pan/Tilt and the PE
%-------------------------------------------------------------------
figure(1);plot(StandPan1,StandPE1,'r*');title('Plot of PE and the Standard Deviations');
xlabel('STD of Parameters');ylabel('STD of the Position Estimation');
```

```
hold on
plot(StandPan2,StandPE2,'r+')
hold on
plot(StandPan3,StandPE3,'r*')
hold on
plot(StandPan4,StandPE4,'r+')
hold on
plot(StandPan5,StandPE5,'r+')
hold on
plot(StandPan6,StandPE6,'r*')
hold on
plot(StandPan7,StandPE7,'r+')
hold on
plot(StandPan8,StandPE8,'r*')
hold on
plot(StandPan9,StandPE9,'r+')
hold on
plot(StandPan10,StandPE10,'r*')
hold on
%---------------------------------------------------------------------
%Plot the Standard Deviations of Phi and the PE
%---------------------------------------------------------------------
plot(StandPhi11,StandPE11,'b*')
hold on
plot(StandPhi12,StandPE12,'b+')
hold on
plot(StandPhi13,StandPE13,'b*')
hold on
plot(StandPhi14,StandPE14,'b+')
hold on
plot(StandPhi15,StandPE15,'b+')
hold on
plot(StandPhi16,StandPE6,'b*')
hold on
plot(StandPhi17,StandPE17,'b+')
hold on
plot(StandPhi18,StandPE18,'b*')
hold on
plot(StandPhi19,StandPE19,'b+')
hold on
plot(StandPhi20,StandPE20,'b*')
hold on
%---------------------------------------------------------------------
%Plot the Standard Deviations of Theta and the PE
%---------------------------------------------------------------------
plot(StandTh21,StandPE21,'g*')
hold on
plot(StandTh22,StandPE22,'g+')
hold on
plot(StandTh23,StandPE23,'g*')
hold on
plot(StandTh24,StandPE24,'g+')
hold on
plot(StandTh25,StandPE25,'g+')
hold on
plot(StandTh26,StandPE26,'g*')
```

```
hold on
plot(StandTh27,StandPE27,'g+')
hold on
plot(StandTh28,StandPE28,'g*')
hold on
plot(StandTh29,StandPE29,'g+')
hold on
plot(StandTh30,StandPE30,'g*')
hold on
%------------------------------------------------------------------
%Plot the Standard Deviations of Psi and the PE
%------------------------------------------------------------------
plot(StandPsi31,StandPE1,'k*')
hold on
plot(StandPsi32,StandPE32,'k+')
hold on
plot(StandPsi33,StandPE3,'k*')
hold on
plot(StandPsi34,StandPE34,'k+')
hold on
plot(StandPsi35,StandPE35,'k*')
hold on
plot(StandPsi36,StandPE36,'k+')
hold on
plot(StandPsi37,StandPE37,'k*')
hold on
plot(StandPsi38,StandPE38,'k+')
hold on
plot(StandPsi39,StandPE39,'k*')
hold on
plot(StandPsi40,StandPE40,'k+')
hold on
%------------------------------------------------------------------
%Plot the Standard Deviations of the P and the PE
%------------------------------------------------------------------
plot(StandP41,StandPE41,'m*')
hold on
plot(StandP42,StandPE42,'m+')
hold on
plot(StandP43,StandPE43,'m*')
hold on
plot(StandP44,StandPE44,'m+')
hold on
plot(StandP45,StandPE45,'m+')
hold on
plot(StandP46,StandPE46,'m*')
hold on
plot(StandP47,StandPE47,'m+')
hold on
plot(StandP48,StandPE48,'m*')
hold on
plot(StandP49,StandPE49,'m+')
hold on
plot(StandP50,StandPE50,'m*')
hold on
%------------------------------------------------------------------
```

```
%Plot the Standard Deviations of the Q and the PE
%----------------------------------------------------------------------
plot(StandQ51,StandPE51,'c*')
hold on
plot(StandQ52,StandPE52,'c+')
hold on
plot(StandQ53,StandPE53,'c*')
hold on
plot(StandQ54,StandPE54,'c+')
hold on
plot(StandQ55,StandPE55,'c+')
hold on
plot(StandQ56,StandPE56,'c*')
hold on
plot(StandQ57,StandPE57,'c+')
hold on
plot(StandQ58,StandPE58,'c*')
hold on
plot(StandQ59,StandPE59,'c+')
hold on
plot(StandQ60,StandPE60,'c*')
hold on
%----------------------------------------------------------------------
%Plot the Standard Deviations of the R and the PE
%----------------------------------------------------------------------
plot(StandR61,StandPE61,'y*')
hold on
plot(StandR62,StandPE62,'y+')
hold on
plot(StandR63,StandPE63,'y*')
hold on
plot(StandR64,StandPE64,'y+')
hold on
plot(StandR65,StandPE65,'y+')
hold on
plot(StandR66,StandPE66,'y*')
hold on
plot(StandR67,StandPE67,'y+')
hold on
plot(StandR68,StandPE68,'y*')
hold on
plot(StandR69,StandPE69,'y+')
hold on
plot(StandR70,StandPE70,'y*')
hold on
```

Note: This m-file can only be run after all eight individual parameter m-files have been properly run. After they are run, all the .mat files created must be opened so that this script file can call the needed variables from the Matlab workspace.

C. TARGET-TRACKING ROUTINE

```
%==========================================================================
%%%%%Target-Tracking Simulation of the NPS UAV
%%%%%by ENS Todd Trago
```

```
%=================================================================
%%This part of the script will run a new part of my Monte Carlo simulation.
%%This simulation will be slightly different than the previous one.
%%The basic idea is the same however, an error is implemented into the
%%system, and analysis will be ran on certain model performances.  In this
%%new simulation random error signals will not be sent to the system via
%%random number generators as before.  This time a pulse width (PW) block
%%will be manipulated in the Simulink model Sim2.  This pulse width block
%%simulates whether or not target tracking is on or off.  When the pulse
%%value of 1 is given, target tracking is on.  When the pulse value of 0 is
%%given, target tracking is off.  This simulation will change the pulse
%%width(% of the period) parameter, and observe how this change will affect
%%the given modal performances.
%-----------------------------------------------------------------
%In order for this simulation to work correctly, you must turn off all of
%the manual switches that were used in the previous part of the Monte Carlo
%simulation.  Then the manual switch that enables the pulse width generator
%in Sim2 must be changed to on.
%-----------------------------------------------------------------
%Iteration one (PW-15%)
%Set the pulse width parameter to 15% manually
%Run the simulation with target-loss
[t]=sim('Sim3',[0 100]);
%Name the parameters that are needed to be saved.
T_Range1=T_Range;
one1=one;
two1=two;
%Save Simulation results in a .mat file
%Savefile= 'PW1.mat';
%Also save all input parameters
save('PW1','t','T_Range1','one1','two1','Tracker');
figure(1)
subplot(2,1,1);plot(t,Tracker,'r');legend('Target-Tracking Sigal','Location','Northeast');
title('Plot Showing the Status of Target-Tracking');
subplot(2,1,2);plot(t,one1,'r',t,T_Range1,'b');legend('one1','True Range1','Location','Northeast');
title('Plot of the Range Estimation');
Hold on
clear all
%-----------------------------------------------------------------
%Iteration two (PW-45%)
%Set the pulse width parameter to 45% manually
%Run the simulation with target-loss
[t]=sim('Sim3',[0 200]);
%Name the parameters that are needed to be saved.
T_Range2=T_Range;
one2=one;
two2=two;
%Save Simulation results in a .mat file
%Savefile= 'PW2.mat';
%Also save all input parameters
save('PW2','t','T_Range2','one2','two2','Tracker');
figure(2)
subplot(2,1,1);plot(t,Tracker,'r');legend('Target-Tracking Sigal','Location','Northeast');
title('Plot Showing the Status of Target-Tracking');
subplot(2,1,2);plot(t,one2,'r',t,T_Range2,'b');legend('one2','True Range2','Location','Northeast');
title('Plot of the Range Estimation');
```

```
Hold on
clear all
%-------------------------------------------------------------------
%Iteration three (PW-75%)
%Set the pulse width parameter to 75% manually
%Run the simulation with target-loss
[t]=sim('Sim3',[0 2000]);
%Name the parameters that are needed to be saved.
T_Range3=T_Range;
one3=one;
two3=two;
%Save Simulation results in a .mat file
%Savefile= 'PW3.mat';
%Also save all input parameters
save('PW3','t','T_Range3','one3','two3','Tracker');
figure(3)
subplot(2,1,1);plot(t,Tracker,'r');legend('Target-Tracking Sigal','Location','Northeast');
title('Plot Showing the Status of Target-Tracking');
subplot(2,1,2);plot(t,one3,'r',t,T_Range3,'b');legend('one3','True Range3','Location','Northeast');
title('Plot of the Range Estimation');
Hold on
clear all
%-------------------------------------------------------------------
%=================================================================
```

D. FULL NOISE ROUTINES

1. Full Noise Simulation M-File

```
%==========================================================
%%%%%Full Noise Scenario of the NPS SUAV
%%%%%by ENS Todd Trago
%==========================================================
%%%%This m-file only deals with all the parameters
%%%%from Sim1.mdl
%%%The file Sim1.mdl must be opened and in the current directory
%%%for this to work correctly
%==========================================================
%-------------------------------------------------------------------
%%This simulation is to be run after the ideal and individual simulations
%%have been run.  It calculates Etta (the Posistion Estimation (PE) angle),
%%and plots it against the total PE error.  This is done so that one can see
%%the trend that noise introduced into the system will have on the
%%performance of the PE system.
%-------------------------------------------------------------------
%%First iteration
%Set the paramters for the 1st part of the sim
set_param('Sim3/UAV 6DOF simulation/Gimballed Camera model/MC Sim1/Random
Number2','Variance','0.001','Mean','0.001');
set_param('Sim3/UAV 6DOF simulation/Gimballed Camera model/MC Sim1/Random
Number3','Variance','0.001','Mean','0.001');
set_param('Sim3/UAV 6DOF simulation/CurGuid Controller/MC Sim2/Random
Number','Variance','0.001','Mean','0.001');
set_param('Sim3/UAV 6DOF simulation/CurGuid Controller/MC Sim2/Random
Number1','Variance','0.001','Mean','0.001');
```

```
set_param('Sim3/UAV 6DOF simulation/CurGuid Controller/MC Sim2/Random
Number2','Variance','0.001','Mean','0.001');
set_param('Sim3/UAV 6DOF simulation/CurGuid Controller/MC Sim3/Random
Number3','Variance','0.001','Mean','0.001');
set_param('Sim3/UAV 6DOF simulation/CurGuid Controller/MC Sim3/Random
Number4','Variance','0.001','Mean','0.001');
set_param('Sim3/UAV 6DOF simulation/CurGuid Controller/MC Sim3/Random
Number5','Variance','0.001','Mean','0.001');
set_param('Sim3/UAV 6DOF simulation/CurGuid Controller/MC Sim4/Random
Number6','Variance','0.001','Mean','0.001');
%Run the simulation with the starting Mean/Variance
[t]=sim('Sim3',[0 200]);
%Set up parameters to be saved
Etta1=Etta;
Conv1=Conv;
Err1= ErrPan+ErrTilt+ErrPhi+ErrTheta+ErrPsi+ErrP+ErrQ+ErrR;
time1=time;
t1=min(time1);
t2=max(time1);
%Find the RMS Value of the Total Error and Etta
y1= sqrt(sum((Err1).*conj(Err1))/size((Err1),1));
y2= sqrt(sum((Etta1).*conj(Etta1))/size((Etta1),1));
y3= sqrt(sum((Conv1).*conj(Conv1))/size((Conv1),1));
%Find the Standard Deviations and Covariances of Etta and Total Error
StandEtta1=std(Etta1);
StandErr1=std(Err1);
CovEtta1=cov(Etta1);
CovErr1=cov(Err1);
StandConv1=std(Conv1);
CovConv1=cov(Conv1);
%Save Simulation results in a .mat file
%Savefile= 'Etta1.mat';
%Also save all input parameters
save('Etta1','time','Etta1','Err1','StandEtta1','StandErr1','CovEtta1','CovErr1','Conv1','StandConv1','
CovConv1','y1','y2','y3','time1','t1','t2');
clear all
%------------------------------------------------------------------
%%Second iteration
%Set the paramters for the 2nd part of the sim
set_param('Sim3/UAV 6DOF simulation/Gimballed Camera model/MC Sim1/Random
Number2','Variance','0.002','Mean','0.002');
set_param('Sim3/UAV 6DOF simulation/Gimballed Camera model/MC Sim1/Random
Number3','Variance','0.002','Mean','0.002');
set_param('Sim3/UAV 6DOF simulation/CurGuid Controller/MC Sim2/Random
Number','Variance','0.002','Mean','0.002');
set_param('Sim3/UAV 6DOF simulation/CurGuid Controller/MC Sim2/Random
Number1','Variance','0.002','Mean','0.002');
set_param('Sim3/UAV 6DOF simulation/CurGuid Controller/MC Sim2/Random
Number2','Variance','0.002','Mean','0.002');
set_param('Sim3/UAV 6DOF simulation/CurGuid Controller/MC Sim3/Random
Number3','Variance','0.002','Mean','0.002');
set_param('Sim3/UAV 6DOF simulation/CurGuid Controller/MC Sim3/Random
Number4','Variance','0.002','Mean','0.002');
set_param('Sim3/UAV 6DOF simulation/CurGuid Controller/MC Sim3/Random
Number5','Variance','0.002','Mean','0.002');
```

```
set_param('Sim3/UAV 6DOF simulation/CurGuid Controller/MC Sim4/Random
Number6','Variance','0.002','Mean','0.002');
%Run the simulation with the starting Mean/Variance
[t]=sim('Sim3',[0 200]);
%Set up parameters to be saved
Etta2=Etta;
Conv2=Conv;
Err2= ErrPan+ErrTilt+ErrPhi+ErrTheta+ErrPsi+ErrP+ErrQ+ErrR;
time2=time;
t3=min(time2);
t4=max(time2);
%Find the RMS Value of the Total Error and Etta
y4= sqrt(sum((Err2).*conj(Err2))/size((Err2),1));
y5= sqrt(sum((Etta2).*conj(Etta2))/size((Etta2),1));
y6= sqrt(sum((Conv2).*conj(Conv2))/size((Conv2),1));
%Find the Standard Deviations and Covariances of Etta and Total Error
StandEtta2=std(Etta2);
StandErr2=std(Err2);
CovEtta2=cov(Etta2);
CovErr2=cov(Err2);
StandConv2=std(Conv2);
CovConv2=cov(Conv2);
%Save Simulation results in a .mat file
%Savefile= 'Etta2.mat';
%Also save all input parameters
save('Etta2','t','Etta2','Err2','StandEtta2','StandErr2','CovEtta2','CovErr2','Conv2','StandConv2','Co
vConv2','y4','y5','y6','time2','t3','t4');
clear all
%-------------------------------------------------------------------
%%Third iteration
%Set the paramters for the 3rd part of the sim
set_param('Sim3/UAV 6DOF simulation/Gimballed Camera model/MC Sim1/Random
Number2','Variance','0.003','Mean','0.003');
set_param('Sim3/UAV 6DOF simulation/Gimballed Camera model/MC Sim1/Random
Number3','Variance','0.003','Mean','0.003');
set_param('Sim3/UAV 6DOF simulation/CurGuid Controller/MC Sim2/Random
Number','Variance','0.003','Mean','0.003');
set_param('Sim3/UAV 6DOF simulation/CurGuid Controller/MC Sim2/Random
Number1','Variance','0.003','Mean','0.003');
set_param('Sim3/UAV 6DOF simulation/CurGuid Controller/MC Sim2/Random
Number2','Variance','0.003','Mean','0.003');
set_param('Sim3/UAV 6DOF simulation/CurGuid Controller/MC Sim3/Random
Number3','Variance','0.003','Mean','0.003');
set_param('Sim3/UAV 6DOF simulation/CurGuid Controller/MC Sim3/Random
Number4','Variance','0.003','Mean','0.003');
set_param('Sim3/UAV 6DOF simulation/CurGuid Controller/MC Sim3/Random
Number5','Variance','0.003','Mean','0.003');
set_param('Sim3/UAV 6DOF simulation/CurGuid Controller/MC Sim4/Random
Number6','Variance','0.003','Mean','0.003');
%Run the simulation with the starting Mean/Variance
[t]=sim('Sim3',[0 200]);
%Set up parameters to be saved
Etta3=Etta;
Conv3=Conv;
Err3= ErrPan+ErrTilt+ErrPhi+ErrTheta+ErrPsi+ErrP+ErrQ+ErrR;
time3=time;
```

```
t5=min(time3);
t6=max(time3);
%Find the RMS Value of the Total Error and Etta
y7= sqrt(sum((Err3).*conj(Err3))/size((Err3),1));
y8= sqrt(sum((Etta3).*conj(Etta3))/size((Etta3),1));
y9= sqrt(sum((Conv3).*conj(Conv3))/size((Conv3),1));
%Find the Standard Deviations and Covariances of Etta and Total Error
StandEtta3=std(Etta3);
StandErr3=std(Err3);
CovEtta3=cov(Etta3);
CovErr3=cov(Err3);
StandConv3=std(Conv3);
CovConv3=cov(Conv3);
%Save Simulation results in a .mat file
%Savefile= 'Etta3.mat';
%Also save all input parameters
save('Etta3','time','Etta3','Err3','StandEtta3','StandErr3','CovEtta3','CovErr3','Conv3','StandConv3','
CovConv3','y7','y8','y9','time3','t5','t6');
clear all
%-------------------------------------------------------------------
%%Fourth iteration
%Set the paramters for the 4th part of the sim
set_param('Sim3/UAV 6DOF simulation/Gimballed Camera model/MC Sim1/Random
Number2','Variance','0.004','Mean','0.004');
set_param('Sim3/UAV 6DOF simulation/Gimballed Camera model/MC Sim1/Random
Number3','Variance','0.004','Mean','0.004');
set_param('Sim3/UAV 6DOF simulation/CurGuid Controller/MC Sim2/Random
Number','Variance','0.004','Mean','0.004');
set_param('Sim3/UAV 6DOF simulation/CurGuid Controller/MC Sim2/Random
Number1','Variance','0.004','Mean','0.004');
set_param('Sim3/UAV 6DOF simulation/CurGuid Controller/MC Sim2/Random
Number2','Variance','0.004','Mean','0.004');
set_param('Sim3/UAV 6DOF simulation/CurGuid Controller/MC Sim3/Random
Number3','Variance','0.004','Mean','0.004');
set_param('Sim3/UAV 6DOF simulation/CurGuid Controller/MC Sim3/Random
Number4','Variance','0.004','Mean','0.004');
set_param('Sim3/UAV 6DOF simulation/CurGuid Controller/MC Sim3/Random
Number5','Variance','0.004','Mean','0.004');
set_param('Sim3/UAV 6DOF simulation/CurGuid Controller/MC Sim4/Random
Number6','Variance','0.004','Mean','0.004');
%Run the simulation with the starting Mean/Variance
[t]=sim('Sim3',[0 200]);
%Set up parameters to be saved
Etta4=Etta;
Conv4=Conv;
Err4= ErrPan+ErrTilt+ErrPhi+ErrTheta+ErrPsi+ErrP+ErrQ+ErrR;
time4=time;
t7=min(time4);
t8=max(time4);
%Find the RMS Value of the Total Error and Etta
y10= sqrt(sum((Err4).*conj(Err4))/size((Err4),1));
y11= sqrt(sum((Etta4).*conj(Etta4))/size((Etta4),1));
y12= sqrt(sum((Conv4).*conj(Conv4))/size((Conv4),1));
%Find the Standard Deviations and Covariances of Etta and Total Error
StandEtta4=std(Etta4);
StandErr4=std(Err4);
```

```
CovEtta4=cov(Etta4);
CovErr4=cov(Err4);
StandConv4=std(Conv4);
CovConv4=cov(Conv4);
%Save Simulation results in a .mat file
%Savefile= 'Etta4.mat';
%Also save all input parameters
save('Etta4','time','Etta4','Err4','StandEtta4','StandErr4','CovEtta4','CovErr4','Conv4','StandConv4','
CovConv4','y10','y11','y12','time4','t7','t8');
clear all
%-----------------------------------------------------------------
%%Fifth iteration
%Set the paramters for the 5th part of the sim
set_param('Sim3/UAV 6DOF simulation/Gimballed Camera model/MC Sim1/Random
Number2','Variance','0.005','Mean','0.005');
set_param('Sim3/UAV 6DOF simulation/Gimballed Camera model/MC Sim1/Random
Number3','Variance','0.005','Mean','0.005');
set_param('Sim3/UAV 6DOF simulation/CurGuid Controller/MC Sim2/Random
Number','Variance','0.005','Mean','0.005');
set_param('Sim3/UAV 6DOF simulation/CurGuid Controller/MC Sim2/Random
Number1','Variance','0.005','Mean','0.005');
set_param('Sim3/UAV 6DOF simulation/CurGuid Controller/MC Sim2/Random
Number2','Variance','0.005','Mean','0.005');
set_param('Sim3/UAV 6DOF simulation/CurGuid Controller/MC Sim3/Random
Number3','Variance','0.005','Mean','0.005');
set_param('Sim3/UAV 6DOF simulation/CurGuid Controller/MC Sim3/Random
Number4','Variance','0.005','Mean','0.005');
set_param('Sim3/UAV 6DOF simulation/CurGuid Controller/MC Sim3/Random
Number5','Variance','0.005','Mean','0.005');
set_param('Sim3/UAV 6DOF simulation/CurGuid Controller/MC Sim4/Random
Number6','Variance','0.005','Mean','0.005');
%Run the simulation with the starting Mean/Variance
[t]=sim('Sim3',[0 200]);
%Set up parameters to be saved
Etta5=Etta;
Conv5=Conv;
Err5= ErrPan+ErrTilt+ErrPhi+ErrTheta+ErrPsi+ErrP+ErrQ+ErrR;
time5=time;
t9=min(time5);
t10=max(time5);
%Find the RMS Value of the Total Error and Etta
y13= sqrt(sum((Err5).*conj(Err5))/size((Err5),1));
y14= sqrt(sum((Etta5).*conj(Etta5))/size((Etta5),1));
y15= sqrt(sum((Conv5).*conj(Conv5))/size((Conv5),1));
%Find the Standard Deviations and Covariances of Etta and Total Error
StandEtta5=std(Etta5);
StandErr5=std(Err5);
CovEtta5=cov(Etta5);
CovErr5=cov(Err5);
StandConv5=std(Conv5);
CovConv5=cov(Conv5);
%Save Simulation results in a .mat file
%Savefile= 'Etta5.mat';
%Also save all input parameters
save('Etta5','time','Etta5','Err5','StandEtta5','StandErr5','CovEtta5','CovErr5','Conv5','StandConv5','
CovConv5','y13','y14','y15','time5','t9','t10');
```

```
clear all
%-------------------------------------------------------------------
%%Sixth iteration
%Set the paramters for the 6th part of the sim
set_param('Sim3/UAV 6DOF simulation/Gimballed Camera model/MC Sim1/Random
Number2','Variance','0.006','Mean','0.006');
set_param('Sim3/UAV 6DOF simulation/Gimballed Camera model/MC Sim1/Random
Number3','Variance','0.006','Mean','0.006');
set_param('Sim3/UAV 6DOF simulation/CurGuid Controller/MC Sim2/Random
Number','Variance','0.006','Mean','0.006');
set_param('Sim3/UAV 6DOF simulation/CurGuid Controller/MC Sim2/Random
Number1','Variance','0.006','Mean','0.006');
set_param('Sim3/UAV 6DOF simulation/CurGuid Controller/MC Sim2/Random
Number2','Variance','0.006','Mean','0.006');
set_param('Sim3/UAV 6DOF simulation/CurGuid Controller/MC Sim3/Random
Number3','Variance','0.006','Mean','0.006');
set_param('Sim3/UAV 6DOF simulation/CurGuid Controller/MC Sim3/Random
Number4','Variance','0.006','Mean','0.006');
set_param('Sim3/UAV 6DOF simulation/CurGuid Controller/MC Sim3/Random
Number5','Variance','0.006','Mean','0.006');
set_param('Sim3/UAV 6DOF simulation/CurGuid Controller/MC Sim4/Random
Number6','Variance','0.006','Mean','0.006');
%Run the simulation with the starting Mean/Variance
[t]=sim('Sim3',[0 200]);
%Set up parameters to be saved
Etta6=Etta;
Conv6=Conv;
Err6= ErrPan+ErrTilt+ErrPhi+ErrTheta+ErrPsi+ErrP+ErrQ+ErrR;
time6=time;
t11=min(time6);
t12=max(time6);
%Find the RMS Value of the Total Error and Etta
y16= sqrt(sum((Err6).*conj(Err6))/size((Err6),1));
y17= sqrt(sum((Etta6).*conj(Etta6))/size((Etta6),1));
y18= sqrt(sum((Conv6).*conj(Conv6))/size((Conv6),1));
%Find the Standard Deviations and Covariances of Etta and Total Error
StandEtta6=std(Etta6);
StandErr6=std(Err6);
CovEtta6=cov(Etta6);
CovErr6=cov(Err6);
StandConv6=std(Conv6);
CovConv6=cov(Conv6);
%Save Simulation results in a .mat file
%Savefile= 'Etta6.mat';
%Also save all input parameters
save('Etta6','time','Etta6','Err6','StandEtta6','StandErr6','CovEtta6','CovErr6','Conv6','StandConv6','
CovConv6','y16','y17','y18','time6','t11','t12');
clear all
%-------------------------------------------------------------------
%%Seventh iteration
%Set the paramters for the 7th part of the sim
set_param('Sim3/UAV 6DOF simulation/Gimballed Camera model/MC Sim1/Random
Number2','Variance','0.007','Mean','0.007');
set_param('Sim3/UAV 6DOF simulation/Gimballed Camera model/MC Sim1/Random
Number3','Variance','0.007','Mean','0.007');
```

```
set_param('Sim3/UAV 6DOF simulation/CurGuid Controller/MC Sim2/Random
Number','Variance','0.007','Mean','0.007');
set_param('Sim3/UAV 6DOF simulation/CurGuid Controller/MC Sim2/Random
Number1','Variance','0.007','Mean','0.007');
set_param('Sim3/UAV 6DOF simulation/CurGuid Controller/MC Sim2/Random
Number2','Variance','0.007','Mean','0.007');
set_param('Sim3/UAV 6DOF simulation/CurGuid Controller/MC Sim3/Random
Number3','Variance','0.007','Mean','0.007');
set_param('Sim3/UAV 6DOF simulation/CurGuid Controller/MC Sim3/Random
Number4','Variance','0.007','Mean','0.007');
set_param('Sim3/UAV 6DOF simulation/CurGuid Controller/MC Sim3/Random
Number5','Variance','0.007','Mean','0.007');
set_param('Sim3/UAV 6DOF simulation/CurGuid Controller/MC Sim4/Random
Number6','Variance','0.007','Mean','0.007');
%Run the simulation with the starting Mean/Variance
[t]=sim('Sim3',[0 200]);
%Set up parameters to be saved
Etta7=Etta;
Conv7=Conv;
Err7= ErrPan+ErrTilt+ErrPhi+ErrTheta+ErrPsi+ErrP+ErrQ+ErrR;
time7=time;
t13=min(time7);
t14=max(time7);
%Find the RMS Value of the Total Error and Etta
y19= sqrt(sum((Err7).*conj(Err7))/size((Err7),1));
y20= sqrt(sum((Etta7).*conj(Etta7))/size((Etta7),1));
y21= sqrt(sum((Conv7).*conj(Conv7))/size((Conv7),1));
%Find the Standard Deviations and Covariances of Etta and Total Error
StandEtta7=std(Etta7);
StandErr7=std(Err7);
CovEtta7=cov(Etta7);
CovErr7=cov(Err7);
StandConv7=std(Conv7);
CovConv7=cov(Conv7);
%Save Simulation results in a .mat file
%Savefile= 'Etta7.mat';
%Also save all input parameters
save('Etta7','time','Etta7','Err7','StandEtta7','StandErr7','CovEtta7','CovErr7','Conv7','StandConv7','
CovConv7','y19','y20','y21','time7','t13','t14');
clear all
%-------------------------------------------------------------------
%%Eigth iteration
%Set the paramters for the 8th part of the sim
set_param('Sim3/UAV 6DOF simulation/Gimballed Camera model/MC Sim1/Random
Number2','Variance','0.008','Mean','0.008');
set_param('Sim3/UAV 6DOF simulation/Gimballed Camera model/MC Sim1/Random
Number3','Variance','0.008','Mean','0.008');
set_param('Sim3/UAV 6DOF simulation/CurGuid Controller/MC Sim2/Random
Number','Variance','0.008','Mean','0.008');
set_param('Sim3/UAV 6DOF simulation/CurGuid Controller/MC Sim2/Random
Number1','Variance','0.008','Mean','0.008');
set_param('Sim3/UAV 6DOF simulation/CurGuid Controller/MC Sim2/Random
Number2','Variance','0.008','Mean','0.008');
set_param('Sim3/UAV 6DOF simulation/CurGuid Controller/MC Sim3/Random
Number3','Variance','0.008','Mean','0.008');
```

```
set_param('Sim3/UAV 6DOF simulation/CurGuid Controller/MC Sim3/Random
Number4','Variance','0.008','Mean','0.008');
set_param('Sim3/UAV 6DOF simulation/CurGuid Controller/MC Sim3/Random
Number5','Variance','0.008','Mean','0.008');
set_param('Sim3/UAV 6DOF simulation/CurGuid Controller/MC Sim4/Random
Number6','Variance','0.008','Mean','0.008');
%Run the simulation with the starting Mean/Variance
[t]=sim('Sim3',[0 200]);
%Set up parameters to be saved
Etta8=Etta;
Conv8=Conv;
Err8= ErrPan+ErrTilt+ErrPhi+ErrTheta+ErrPsi+ErrP+ErrQ+ErrR;
time8=time;
t15=min(time8);
t16=max(time8);
%Find the RMS Value of the Total Error and Etta
y22= sqrt(sum((Err8).*conj(Err8))/size((Err8),1));
y23= sqrt(sum((Etta8).*conj(Etta8))/size((Etta8),1));
y24= sqrt(sum((Conv8).*conj(Conv8))/size((Conv8),1));
%Find the Standard Deviations and Covariances of Etta and Total Error
StandEtta8=std(Etta8);
StandErr8=std(Err8);
CovEtta8=cov(Etta8);
CovErr8=cov(Err8);
StandConv8=std(Conv8);
CovConv8=cov(Conv8);
%Save Simulation results in a .mat file
%Savefile= 'Etta8.mat';
%Also save all input parameters
save('Etta8','time','Etta8','Err8','StandEtta8','StandErr8','CovEtta8','CovErr8','Conv8','StandConv8','
CovConv8','y22','y23','y24','time8','t15','t16');
clear all
%--------------------------------------------------------------------
%%Ninth iteration
%Set the paramters for the 9th part of the sim
set_param('Sim3/UAV 6DOF simulation/Gimballed Camera model/MC Sim1/Random
Number2','Variance','0.009','Mean','0.009');
set_param('Sim3/UAV 6DOF simulation/Gimballed Camera model/MC Sim1/Random
Number3','Variance','0.009','Mean','0.009');
set_param('Sim3/UAV 6DOF simulation/CurGuid Controller/MC Sim2/Random
Number','Variance','0.009','Mean','0.009');
set_param('Sim3/UAV 6DOF simulation/CurGuid Controller/MC Sim2/Random
Number1','Variance','0.009','Mean','0.009');
set_param('Sim3/UAV 6DOF simulation/CurGuid Controller/MC Sim2/Random
Number2','Variance','0.009','Mean','0.009');
set_param('Sim3/UAV 6DOF simulation/CurGuid Controller/MC Sim3/Random
Number3','Variance','0.009','Mean','0.009');
set_param('Sim3/UAV 6DOF simulation/CurGuid Controller/MC Sim3/Random
Number4','Variance','0.009','Mean','0.009');
set_param('Sim3/UAV 6DOF simulation/CurGuid Controller/MC Sim3/Random
Number5','Variance','0.009','Mean','0.009');
set_param('Sim3/UAV 6DOF simulation/CurGuid Controller/MC Sim4/Random
Number6','Variance','0.009','Mean','0.009');
%Run the simulation with the starting Mean/Variance
[t]=sim('Sim3',[0 200]);
%Set up parameters to be saved
```

```
Etta9=Etta;
Conv9=Conv;
Err9= ErrPan+ErrTilt+ErrPhi+ErrTheta+ErrPsi+ErrP+ErrQ+ErrR;
time9=time;
t17=min(time9);
t18=max(time9);
%Find the RMS Value of the Total Error and Etta
y25= sqrt(sum((Err9).*conj(Err9))/size((Err9),1));
y26= sqrt(sum((Etta9).*conj(Etta9))/size((Etta9),1));
y27= sqrt(sum((Conv9).*conj(Conv9))/size((Conv9),1));
%Find the Standard Deviations and Covariances of Etta and Total Error
StandEtta9=std(Etta9);
StandErr9=std(Err9);
CovEtta9=cov(Etta9);
CovErr9=cov(Err9);
StandConv9=std(Conv9);
CovConv9=cov(Conv9);
%Save Simulation results in a .mat file
%Savefile= 'Etta9.mat';
%Also save all input parameters
save('Etta9','time','Etta9','Err9','StandEtta9','StandErr9','CovEtta9','CovErr9','Conv9','StandConv9','
CovConv9','y25','y26','y27','time9','t17','t18');
clear all
%-------------------------------------------------------------------
%%Tenth iteration
%Set the paramters for the 10th part of the sim
set_param('Sim3/UAV 6DOF simulation/Gimballed Camera model/MC Sim1/Random
Number2','Variance','0.01','Mean','0.01');
set_param('Sim3/UAV 6DOF simulation/Gimballed Camera model/MC Sim1/Random
Number3','Variance','0.01','Mean','0.01');
set_param('Sim3/UAV 6DOF simulation/CurGuid Controller/MC Sim2/Random
Number','Variance','0.01','Mean','0.01');
set_param('Sim3/UAV 6DOF simulation/CurGuid Controller/MC Sim2/Random
Number1','Variance','0.01','Mean','0.01');
set_param('Sim3/UAV 6DOF simulation/CurGuid Controller/MC Sim2/Random
Number2','Variance','0.01','Mean','0.01');
set_param('Sim3/UAV 6DOF simulation/CurGuid Controller/MC Sim3/Random
Number3','Variance','0.01','Mean','0.01');
set_param('Sim3/UAV 6DOF simulation/CurGuid Controller/MC Sim3/Random
Number4','Variance','0.01','Mean','0.01');
set_param('Sim3/UAV 6DOF simulation/CurGuid Controller/MC Sim3/Random
Number5','Variance','0.01','Mean','0.01');
set_param('Sim3/UAV 6DOF simulation/CurGuid Controller/MC Sim4/Random
Number6','Variance','0.01','Mean','0.01');
%Run the simulation with the starting Mean/Variance
[t]=sim('Sim3',[0 200]);
%Set up parameters to be saved
Etta10=Etta;
Conv10=Conv;
Err10= ErrPan+ErrTilt+ErrPhi+ErrTheta+ErrPsi+ErrP+ErrQ+ErrR;
time10=time;
t19=min(time10);
t20=max(time10);
%Find the RMS Value of the Total Error and Etta
y28= sqrt(sum((Err10).*conj(Err10))/size((Err10),1));
y29= sqrt(sum((Etta10).*conj(Etta10))/size((Etta10),1));
```

```
y30= sqrt(sum((Conv10).*conj(Conv10))/size((Conv10),1));
%Find the Standard Deviations and Covariances of Etta and Total Error
StandEtta10=std(Etta10);
StandErr10=std(Err10);
CovEtta10=cov(Etta10);
CovErr10=cov(Err10);
StandConv10=std(Conv10);
CovConv10=cov(Conv10);
%Save Simulation results in a .mat file
%Savefile= 'Etta10.mat';
%Also save all input parameters
save('Etta10','time','Etta10','Err10','StandEtta10','StandErr10','CovEtta10','CovErr10','Conv10','Sta
ndConv10','CovConv10','y28','y29','y30','time10','t19','t20');
clear all
%==============================================================
```

2. Eta Sensitivity M-File

```
%==============================================================
%%%%%Final Eta vs. Total Error Plots
%%%%%by ENS Todd Trago
%==============================================================
%%This script file produces the final plots for the Etta parameter.  It can
%%only be run after the file Etta.m has be successfully run.  This script
%%file creates three plots:  RMS values of Etta vs. Total PE Error,
%%Standard Deviation values of Etta vs. Total PE Error, and Covariances
%Etta vs. Total PE Error.
%-------------------------------------------------------------------
%Plot the RMS values of Eta vs. the Total PE Error
%-------------------------------------------------------------------
figure(1);plot(y2,y1,'bl*');title('Plot of the RMS values of Etta vs. Total Error');
xlabel('Etta RMS');ylabel('Total Error RMS');
hold on
plot(y5,y4,'bl+')
hold on
plot(y8,y7,'bl*')
hold on
plot(y11,y10,'bl+')
hold on
plot(y14,y13,'bl*')
hold on
plot(y17,y16,'bl+')
hold on
plot(y20,y19,'bl*')
hold on
plot(y23,y22,'bl+')
hold on
plot(y26,y25,'bl+')
hold on
plot(y29,y28,'bl+')
%-------------------------------------------------------------------
%Plot the Standard Deviations of Eta vs. Total PE Error
%-------------------------------------------------------------------
figure(2);plot(StandEtta1,StandErr1,'r*');title('Plot of Total PE Error and Etta Standard
Deviations');
```

```
xlabel('Etta STD');ylabel('Total PE Error STD');
hold on
plot(StandEtta2,StandErr2,'r+')
hold on
plot(StandEtta3,StandErr3,'r*')
hold on
plot(StandEtta4,StandErr4,'r+')
hold on
plot(StandEtta4,StandErr4,'r*')
hold on
plot(StandEtta5,StandErr5,'r+')
hold on
plot(StandEtta6,StandErr6,'r*')
hold on
plot(StandEtta7,StandErr7,'r+')
hold on
plot(StandEtta8,StandErr8,'r*')
hold on
plot(StandEtta9,StandErr9,'r+')
hold on
plot(StandEtta10,StandErr10,'r*')
%-----------------------------------------------------------------
%Plot the Covariances of Eta and the PE Error
%-----------------------------------------------------------------
figure(3);plot(CovEtta1,CovErr1,'b*');title('Plot of Total PE Error and Etta Covariances');
xlabel('Etta COV');ylabel('Total PE Error COV');
hold on
plot(CovEtta2,CovErr2,'b+')
hold on
plot(CovEtta3,CovErr3,'b*')
hold on
plot(CovEtta4,CovErr4,'b+')
hold on
plot(CovEtta5,CovErr5,'b*')
hold on
plot(CovEtta6,CovErr6,'b+')
hold on
plot(CovEtta7,CovErr7,'b*')
hold on
plot(CovEtta8,CovErr8,'b+')
hold on
plot(CovEtta9,CovErr9,'b*')
hold on
plot(CovEtta10,CovErr10,'b+')
%===========================================================
```

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF REFERENCES

1. Prince, Robert A. *Autonomous Visual Tracking of Stationary Targets Using Small Unmanned Aerial Vehicles*. Masters Thesis, Naval Postgraduate School, Monterey, CA, 2004.
2. Lizarraga, Mariano. *Autonomous Landing System for a UAV*. Masters Thesis, Naval Postgraduate School, Monterey, CA, 2004.
3. Rubinstein, Reuven Y. *Simulation and the Monte Carlo Method*. New York: John Wiley & Sons Inc., 1981.
4. Kalos, Malvin H., and Whitlock, Paula A. *Monte Carlo Methods, Volume I: Basics*. John Wiley & Sons Inc., 1986.
5. Spiegel, Murray R. *Theory and Problems of Probability and Statistics*. McGraw-Hill Inc., 1975.
6. Vagilienti, Bill and Ross Hoag. *Piccolo System User Guide*. Cloud Cap Technologies, 2004.
7. Parkinson, Bradford W., and Spilker Jr., James J. *Global Positioning System: Theory and Applications*. American Institute of Aeronautics and Astronautics, Inc., 1996.

THIS PAGE INTENTIONALLY LEFT BLANK

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California

3. Dr. Isaac Kaminer
   Naval Postgraduate School
   Monterey, CA

4. Dr. Vladimir Dobrokhodov
   Naval Postgraduate School
   Monterey, CA

5. Dr. Anthony Healey
   Naval Postgraduate School
   Monterey, CA